МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра прикладної математики та моделювання складних систем

Допущено до захисту
Завідувач кафедри ПМ та МСС
_____ Коплик І.В.

«__»_____20___р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»

спеціальність 113 «Прикладна математика»
освітньо-професійна програма «Наука про дані та моделювання складних систем»

тема роботи «**МОДЕЛЮВАННЯ УЗАГАЛЬНЕНОЇ ЗАДАЧІ ПРО ПРИЗНАЧЕННЯ З ПІДБОРОМ ЕФЕКТИВНОГО АЛГОРИТМУ**»

Виконавець

студент факультету ЕлІТ

Болмат Олександр Володимирович _____

Науковий керівник

старший викладач, кандидат фіз.-мат. наук

Сушко Тетяна Сергіївна _____

Суми – 2020

РЕФЕРАТ

Кваліфікаційна робота: 56 с., 13 рисунків, 19 джерел.

Мета роботи: Побудова математичної моделі та відповідного алгоритму розв’язання узагальненої задачі про призначення на прикладі задачі про сканування книг для проекту Google Books.

Об’єкт дослідження: процес сканування книг з обмеженнями на ресурси, загальний час та процеси реєстрації учасників проекту.

Предмет дослідження: математична модель узагальненої задачі про призначення пов’язаної з проблемою сканування книг.

Методи навчання: рекурентні співвідношення, евристичні підходи, моделювання математичної постанови задачі про призначення.

В роботі побудовано моделі для задачі про оптимальне призначення виконавцям (бібліотекам) робіт (сканування книг) за наявності обмежень на ресурси виконавців (час реєстрації для виконання робіт, обмежена кількість робіт на день) та обмеженні загального часу. Задача розв’язується з залученням евристичних алгоритмів, що застосовні для задач великого розміру. Побудовані алгоритми, що ґрунтуються на принципі оптимальності з динамічного програмування, та ітераційних послідовностях перестановок робіт з сортуванням. Наводяться результати програмної реалізації.

УЗАГАЛЬНЕНА ЗАДАЧА ПРО ПРИЗНАЧЕННЯ, ЗАДАЧА КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ, ДЕТЕРМІНОВАНІ КЕРОВАНІ ПРОЦЕСИ, ДИНАМІЧНЕ ПРОГРАМУВАННЯ, МЕТОД МАКА, УГОРСЬКИЙ МЕТОД.

Зміст

Вступ

У сучасному світі часто виникають проблеми розподілу, які можуть бути зображені у вигляді задач про призначення з додатковими вимогами (задача планування вантажоперевезень, задача управління та багато інших). Проблема розподілу виникає тому, що існують ресурси, такі як люди, машини тощо, мають різну ступінь ефективності при виконання різних видів діяльності, а тому витрати або час, що витрачаються на процес, прибуток або збитки від виконання різних видів діяльності різні.

В галузі прикладної математики, зокрема, теорії оптимізації та математичного моделювання високу прикладну цінність мають задачі про призначення. Найбільш складними з точки зору складності алгоритмів вважаються задачі комбінаторної оптимізації. Побудова математичних моделей таких задача, та їх ефективна реалізація є важливим науковим напрямком в області прикладної математики. Популярність задач про призначення, приводить до розвитку різноманітних моделей, що використовують різні розділи прикладної математики: лінійного та нелінійного програмування, теорії керування, теорії графів та мереж, теорії ігор, комп'ютерних наук та ін.

Задача про призначення є окремим випадком транспортної задачі, тому для її вирішення можна скористатися будь-яким алгоритмом лінійного програмування. Однак, основними методами для вирішення задач про призначення вважаються угорський метод та метод Мака. Обидва методи засновані на тому, що положення оптимального рішення не змінюється, якщо до кожного елементу рядка і стовпця додати або відняти одне і те ж значення.

Робота присвячена підбору ефективного алгоритму для вирішення узагальненої задачі про призначення на прикладі завдання про сканування книг.

1 Літературний огляд

1.1 Задача про призначення. Узагальнена задача про призначення

Задача про призначення - це особливий тип задач лінійного програмування, який стосується стосуються питання, як призначити n елементів (наприклад, робочих місць) n машинам (або працівників) найкращим чином (рисунок 1). Він робить це таким чином, що витрати або час, задіяні в процесі, є мінімальними, а прибуток або продаж максимальними. Хоча існують проблеми, які можна вирішити за допомогою симплексного методу або транспортного методу, але модель призначення дає простіший підхід до вирішення цих проблем [1].

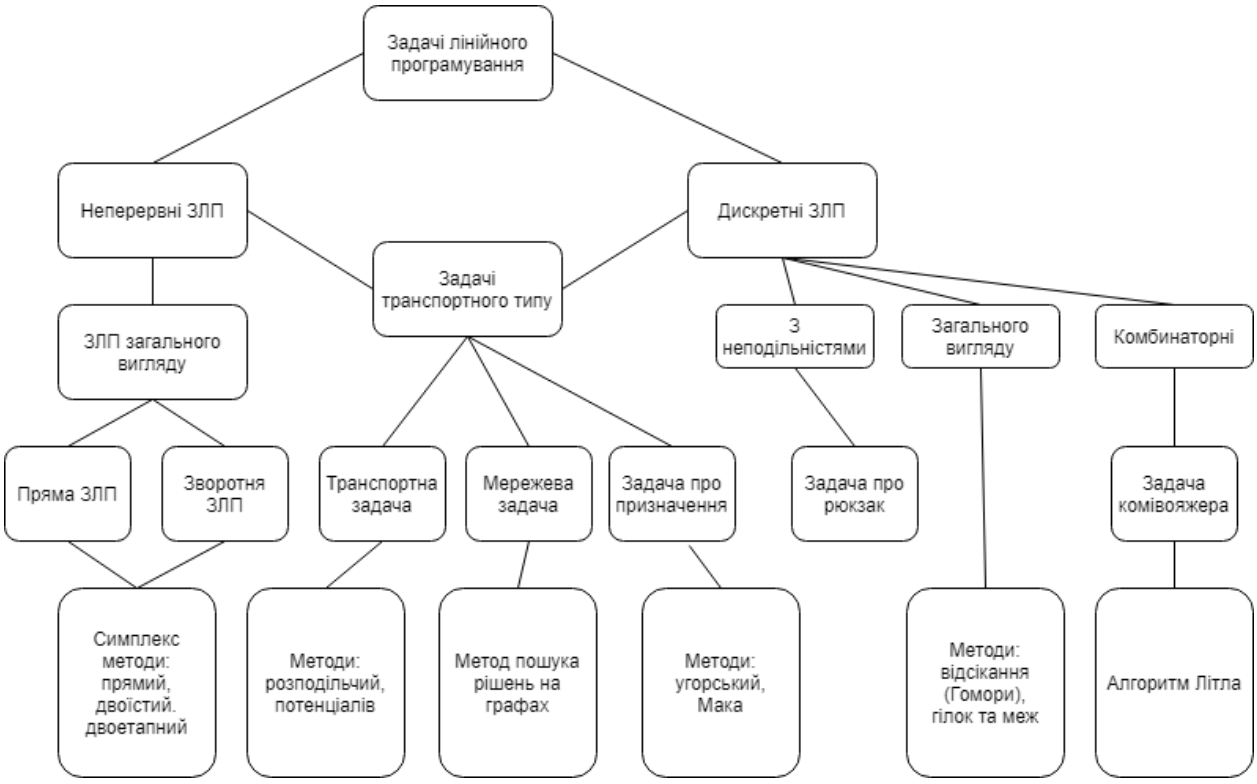

Рис. 1 – Задачі лінійного програмування

Задача про призначення є окремим випадком транспортної задачі, тому для її вирішення можна скористатися будь-яким алгоритмом лінійного програмування. Угорський метод та метод Мака являються основними методами рішень задач про призначення, а саме угорський метод вважається найбільш ефективним.

Саму задачу про призначення можна сформулювати наступним чином.

Заданий перелік робіт $A_1,...,A_m$. Для їх виконання можна використовувати n верстатів (або іншого обладнання) $B_1,...,B_n$, причому одна робота може виконуватися одним верстатом. Виконання роботи A_i на верстаті B_j пов'язано з витратами c_{ij}. Якщо яка-небудь робота A_i не може бути виконана на верстаті B_j, то відповідну величину c_{ij} будемо вважати рівною досить великому числу [2].

Необхідно так розподілити заданий комплекс робіт $\{A_1,...,A_m\}$ за наявним парку обладнання $\{B_1,...,B_n\}$, щоб загальні витрати були мінімальними. Для формалізації цієї задачі введемо змінну x_{ij} рівну 1, якщо i-та робота виконується на j-му верстаті та x_{ij} рівну 0 якщо i-та робота не виконується на j-му верстаті [2].

Тоді задача зводиться до відшукання $n\,m$ чисел x_{ij}, при яких

$$y = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij} \to \min \qquad (1.1)$$

і виконуються умови

$$\forall i:\ \sum_{j=1}^{n} x_{ij} = 1,\ \ i = 1,...,m, \qquad (1.2)$$

$$\forall j:\ \sum_{i=1}^{m} x_{ij} = 1,\ \ j = 1,...,n, \qquad (1.3)$$

$$x \in \{0,1\}. \qquad (1.4)$$

Цю задачу можна вважати транспортною, але для цього необхідно роботи розглядати як вихідні пункти (пункти відправлення), а верстати, як пункти призначення. В результаті рішення кожен пункт відправлення (робота A_i) повинен бути пов'язаний з певним пунктом призначення (B_j). Із змісту задачі, за якою суми, які знаходяться в (1.2; 1.3), повинні бути рівні 1, випливає, що в лівих частинах кожного з рівнянь (1.2 - 1.3) одна із змінних

буде дорівнює 1, а решта - рівні 0 [2, 3].

Також існує узагальнена задача про призначення. Вона відрізняється від звичайної задачі про розподіл тим, що множина завдань та множина виконавців може не співпадати між собою, а також існує обмеження на ресурси робітників [3].

Узагальнена задача присвоєння (Generalized assignment problem) - це проблема мінімізації витрати на присвоєння n різних елементів m робітникам, таким чином, щоб кожен елемент присвоювався точно до одного робітника за умови обмеження їх можливостей [4].

Цю проблему можна сформулювати як:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij} \qquad (1.5)$$

$$\sum_{j=1}^{n} a_{ij} x_{ij} \le b_i,\ i = 1,...,m, \qquad (1.6)$$

$$\sum_{i=1}^{m} x_{ij} = 1,\ \ j = 1,...,n, \qquad (1.7)$$

$$x_{ij} \in \{0,1\}. \qquad (1.8)$$

Тут a_{ij} - вартість присвоєння елемента j робітнику i, a_{ij} - вимога щодо можливостей робітника i за елементом j, і b_i – можливість робітника i.

Узагальнена задача присвоєння та його підструктура виникають у багатьох проблемах, таких як маршрутизація транспортних засобів, розташування об'єкта та багато інших [4].

1.2 Угорський метод вирішення задач про призначення

Угорський метод є спрощеним алгоритмом рішення і заснований на симплекс-методі. Основна ідея угорського методу полягає в переході від вихідної квадратної матриці вартості C до еквівалентної їй матриці Ce з невід'ємними елементами і системою n незалежних нулів, з яких ніякі два не належать одному й тому ж рядку або одному і тому ж стовпчику. Для заданого n існує n! допустимих рішень. Якщо в матриці призначення X розташувати

n одиниць так, що в кожному рядку і стовпці знаходиться тільки по одній одиниці, розставлених відповідно до розташованими n незалежними нулями еквівалентної матриці вартості Ce, то отримаємо допустимі рішення задачі про призначення [5, 6].

Слід мати на увазі, що для будь-якого неприпустимого призначення відповідна йому вартість умовно покладається рівною досить великому числу M в задачах на мінімум. Якщо вихідна матриця не є квадратною, то слід ввести додатково необхідну кількість рядків або стовпців, а їх елементам привласнити значення, які визначаються умовами завдання, а домінуючі альтернативні дорогі або дешеві виключити [6, 7, 8].

Угорського метод поділяється на два види алгоритмів: мінімізації та максимізації.

Алгоритм мінімізації

1. В кожному стовпчику матриці C знаходимо мінімальний елемент, а потім віднімаємо його від усіх елементів стовпчика.

2. В кожному рядку матриці C_0 знаходимо мінімальний елемент, а потім віднімаємо його від усіх елементів рядка.

3. Знаходимо рядок лише з одним нулем та позначаємо його. В стовпчику, де знаходиться цей нуль, закреслюємо решту нулів.

4. Повторюємо крок 3, але тепер шукаємо стовпчик з одним нулем та позначаємо його. Решту нулів закреслюємо.

5. Якщо, по завершенню кроків 3 та 4 залишаються не відмічені нулі, то відмічаємо будь-який з них, а решту закреслюємо (і в стовпчику і в рядку).

6. Якщо кожний рядок і стовпчик містить лише 1 нуль, то отримано оптимальне рішення. В іншому випадку проводиться мінімальна кількість вертикальних та горизонтальних прямих через усі нулі, які перетинаються та знаходимо найменший елемент. Його віднімаємо від усіх не закреслених чисел та повторюємо алгоритм для нової матриці, починаючи з кроку 3 [9, 10].

Алгоритм максимізації

Перший крок.

При максимізації цільової функції в кожному стовпці матриці C знайти максимальний елемент і кожен елемент цього стовпця відняти від максимального. В результаті утворюється матриця C' з невід'ємними елементом (елементами). В кожному стовпці матриці C' є, принаймні, один нуль. Потім у кожному рядку матриці C_0 знайти максимальний елемент і кожен елемент цього рядка відняти від максимального.

В результаті перетворень утворюється матриця C_0 з невід'ємними елементами. У кожному стовпці та рядку матриці C_0 існує, принаймні один нуль. Переходимо до другого кроку.

Другий крок.

Позначаємо довільний нуль в першому стовпці зірочкою. Починаючи з другого стовпчика матриці C_0 відзначати в кожному з них зірочкою нуль, розташований в рядку, де немає нуля із зірочкою. Зірочкою позначається лише по одному нулю в кожному стовпчику. [8, 10]. Якщо серед невиділених стовпців нової матриці C_k не з’явилося нульових елементів, то перейти до кроку номер три.

Якщо ж невиділений нуль матриці C_k з’явився, то для існує два можливих варіанти:

1) в рядку з невиділеним нулем, є і нуль із зірочкою;

2) нуль із зірочкою в цьому рядку відсутній.

У першому випадку позначаємо штрихом невиділений нуль і виділяємо рядок, в якому він знаходиться, додаванням знаку «+» праворуч від нього. Після цього знищуємо знак «+», обводячи його колом над тим стовпчиком, на перетині якого міститься нуль із зірочкою.

Потім шукаємо в ньому не виділений нуль (нулі), що не відмічений зірочкою.

• За умови, що такий нуль є єдиним в стовпчику, відзначити його штрихом і позначити рядок (рядки), що містить такий нуль (нулі), знаком «+». Після чого переглянути цей рядок (рядки), шукаючи в них нуль із зірочкою [6, 8].

• Якщо такий нуль в стовпці знайдений, але він не єдиний в стовпці, то з них потрібно вибрати:

- в першу чергу нуль, в одному рядку з яким відсутній 0^*;

- в другу чергу нуль, в одному рядку з яким присутній 0^*, але в одному стовпці з даним 0^* присутній невиділений нуль;

- в останню чергу нуль, в одному рядку з яким присутній 0^*, але в одному стовпці з цим 0^* відсутній не виділений нуль;

Можливі такі результати роботи цього процесу:

1) Якщо всі нулі матриці C_k виділені, то переходимо до наступного кроку;

2) Якщо є невиділений нуль в рядку, де немає нуля із зіркою, то перейти до попереднього етапу, позначивши штрихом останній за порядком нуль [6].

Третій крок.

Побудувати наступній ланцюжок з елементів матриці C_k: вихідний нуль зі штрихом та нуль із зірочкою (вони розташовані в одному стовпці) та нуль зі штрихом, який розташований в одному рядку з попереднім нулем із зірочкою. Після чого утворюється ланцюжок шляхом пересуванням від $0'$ до 0^* по стовпцю та від 0^* до $0'$ по рядку.

Потім елементи ланцюжка, що стоять на непарних місцях ($0'$), позначити зірочками, знищуючи їх над парними елементами (0^*). Далі знищити всі штрихи над елементами матриці C_k і знаки «+» [8, 11]. Переходимо до заключного четвертого кроку.

Четвертий крок.

За умови, що всі нулі матриці C_k позначені, перебувають у позначених рядках або стовпцях, серед невиділених елементів матриці C_k вибрати мінімальний елемент і позначити його $h > 0$.

Далі слід відняти h з усіх елементів матриці C_k, розташованих у невиділених рядках, і додати h до всіх елементів матриці C_k, розташованим у виділених стовпчиках.

В результаті перетворень отримується нова матриця $C_k^{(1)}$, яка еквівалентна C_k. Оскільки серед невиділених елементів матриці з'являться нові нулі, потрібно повернутися до другого кроку, де буде розглядатися матриця $C_k^{(1)}$ замість матриці C_k.

Завершивши перший етап або перейти до другого етапу, якщо невиділений нуль знаходиться в рядку, яка не містить нуля із зірочкою, або знову повернутися до третього етапу, якщо в результаті виконання першого етапу всі нулі матриці виявляться виділеними. [8, 11].

1.3 Метод Мака

Метод Мака є ітераційним процесом, який заснований на виборі в кожному рядку мінімального елемента. Мінімальні елементи рядків не розташовані в початковому стані у всіх стовпчиках. Оптимальне рішення буде лише тоді, коли в результаті перетворень буде розподілено мінімальні елементи по всіх стовпцях. Матрицю потрібно перетворити таким чином, щоб кожен рядок отримав свій мінімальний елемент [12, 13]. Метод Мака застосовується для розподілу функцій між елементами системи, за умови, коли кожен з цих елементів має здатність виконувати будь-яку з функцій з різною ефективністю [8, 13].

Алгоритм метода Мака

1. Поділити безліч стовпців на A і A', де

A- вибрана множина,

A' - невибрана

Всі стовпці відносяться до A' на початку обчислень.

З множини A' обрати стовпець, який містить більше одного підкресленого елемента та перемістити цей стовпчик з A' до A [6, 14].

Підкреслений елемент буде мінімальним в рядку.

2. За умови, що підкреслений елемент знаходиться в множині A, знайти різницю між мінімальним підкресленим і мінімальним не підкресленим елементами для кожного рядка. Поміж усіх знайдених різниць обрати мінімальну.

3. Збільшити всі елементи матриці A на обрану під час попереднього кроку мінімальну різницю.

4. У рядку, який містить мінімальну різницею відзначити мінімальний не підкреслений елемент $a_i^{'}$ та позначити його пунктиром (1.9).

$$\begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & a_i & \dots & \ddot{a}_i^{'} \\ \dots & \dots & \dots & \dots \\ \dots & a_j & \dots & a_j^{'} \end{bmatrix} \quad \min(a_i^{'} - a_i) = (a_r^{'} - a_r) \quad (1.9)$$

5. Стовпчик, який містить зазначений пунктиром елемент $a_r^{'}$, перенеситься до множини C. Якщо множина C містить більше ніж два не підкреслених елемента, то перенести C з A' до A і повернутися до 2-го кроку. В іншому випадку, перейти до наступного кроку.

6. Підкреслити елемент $a_r^{'}$ зазначений пунктиром.

7. Прибрати підкреслення з вихідного елемент в рядку з мінімальною різницею. Стовпець з елементом a_r позначити D.

8. За умови, коли D не містить інших підкреслених елементів, він має містити елементи, позначені пунктиром. Цей елемент позначити, як $a_r^{'}$ і повернутися до 6-го кроку .

Якщо D містить ще 1 підкреслений елемент, то повністю підкреслені елементи утворюють новий базис. Тоді повернутися до самого початку і провести всі операції знов [13, 14].

1.4 Динамічне програмування. Детерміновані керовані процеси

При постановці задач динамічного програмування використовується поняття “керована динамічна система” та “стан системи” [15]. Нагадаємо їх.

Нехай маємо об’єкт, що змінюється в часі, на який впливає зовнішня дія $u(t)$ (керування), а $x(t)$– деякий опис цього об’єкту в момент часу t. Якщо за відомого керування $u(\tau)$, $\left(\tau \in [t, \bar{t}]\right)$, знаючи опис $x(t)$ в момент часу t, можна однозначно визначити його значення $x(\bar{t})$ для будь-якого $\bar{t} > t$, то такий опис називають *повним*. Повний опис називають *станом*, безліч можливих станів - *простором станів*. Сам об'єкт, що допускає можливість повного опису, називають динамічною системою. Будь-яка динамічна система передбачає наявність однозначного оператора F, що дозволяє по $x(t)$ визначати $x(\bar{t})$ для будь-якого допустимого моменту часу $\bar{t} > t$. Оскільки для керованої системи оператор F залежить від управління, повинні бути задані множини їх можливих значень, що залежать від поточного моменту часу і поточного стану. Таким чином, динамічна система визначається таким набором: $\langle X, T, F, U \rangle$, де X – простір станів, T – множина допустимих моментів часу.

Розглянемо керовану динамічну систему з дискретним часом, стан якої в момент часу k описується вектором n $x_k \in R^n$, а закон зміни її стану визначається співвідношенням, що задає оператор динамічної системи: $x_{k+1} = f_{k+1}(x_k, u_{k+1})$. Управління u_{k+1} вибираються з множин управлінь, допустимих в поточний момент часу k для стану x_k, тобто $u_{k+1} \in U_{k+1}(x_k) \subset R^m$, де множини $U_{k+1}(x_k)$ задані.

Будемо розглядати тільки такі завдання, в яких дискретний час k змінюється в заздалегідь відомих межах: $k = 0,1,..., N-1$, де N задано. Відомо,

що x_0 належить заданій множині початкових станів X_0. Потрібно також, щоб кінцевий стан x_N належав заданій множині X_N допустимих фінальних станів. Кожному переходу зі стану x_k в наступний стан поставимо у відповідність функцію витрат (або доходів) на поточному кроці $d_{k+1}(x_k, u_{k+1})$, залежну, окрім x_k, також від моменту часу і застосовуваного управління. Потрібно знайти такий початковий стан $x_0^* \in X_0$ і такий допустимий набір управлінь $u_1^*, \ldots, u_N^*$, переводить систему в один зі станів $x_N^* \in X_N$, щоб загальні витрати, які є адитивною функцією витрат на окремих кроках, були мінімальні.

Зауважимо, що вихідні формулювання прикладних задач оптимізації часто не відповідають наведеній вище стандартній постановці, хоча після відповідного перетворення можуть бути зведені до неї [16, 17]. Для їх успішного вирішення методами динамічного програмування часто необхідно штучно переформулювати задачу так, щоб вона виявилася записаною через деякий керований динамічний процес. У цьому випадку найбільш важливо правильно ввести поняття стану. Критерієм правильності вибору є дотримання кількох вимог: новий стан, а також функція витрат можуть залежати тільки від попереднього стану, поточного управління та моментів часу; обмеження на поточне управління можуть залежати тільки від попереднього стану і моментів часу. Порушення хоча б одного з правил говорить про неправильний вибір стану або управління.

Для вирішення завдань динамічного програмування використовується підхід, розроблений в лабораторії Р. Беллмана в 50-х роках XX століття [16, 18]. Відповідно до цього підходу, замість вихідної покрокової задачі потрібно розглянути набір допоміжних завдань, подібних вихідної, але включають різну кількість кроків (від 1 до N). У цих завданнях потрібно визначити оптимальні витрати: на одному останньому кроці, двох останніх кроках, трьох, і так далі за умови, що вважається відомим стан, в якому опинився керований динамічний процес перед виконанням цих завершальних кроків. Підхід Р.

Беллмана заснований на тому, що можна встановити рекурентний зв'язок між функціями оптимальних витрат в задачах з кількістю кроків $(k+1)$ і k.

Нехай Y_k – безліч станів, з яких (при використанні допустимих управлінь) можна рівно за k кроків потрапити в одне з станів фінального безлічі X_N. При цьому можна вважати, що $Y_0 = X_N$. Візьмемо довільне $x_{N-k} \in Y_k$ і позначимо через $S_k(x_{N-k})$ функцію, яка описує залежність оптимальних витрат від стану x_{N-k} за k останніх кроків, які переводять систему з x_{N-k} в X_N. Такі функції називають функціями Беллмана.

Оскільки для станів x_{N-1} із множини Y_1 перехід в X_N відбувається за один крок, то функція оптимальних витрат $S_1(x_{N-1})$ може бути записана наступним чином:

$$S_1(x_{N-1}) = min\{d_N(x_{N-1}, u_N)\} \qquad (1.10)$$
$$u_N \in U_N(x_{N-1})$$
$$f_N(x_{N-1}, u_N) \in X_N$$

Зауважимо, що друге обмеження необхідно для того, щоб гарантувати досягнення заданого фінального безлічі X_N. Значення u_N, при якому досягається мінімум в цьому співвідношенні, позначимо через $u_N^*(x_{N-1})$. Можна довести, що функція оптимальних витрат $S_{k+1}(x_{N-k-1})$ кожної наступної задачі рекурентно виражається через попередню $S_k(x_{N-k})$, а саме, для $x_{N-k-1} \in Y_{k+1}$:

$$S_{k+1}(x_{N-k-1}) = min\left\{d_{N-k}(x_{N-k-1}, u_{N-k}) + S_k\left(f_{N-k}(x_{N-k-1}, u_{N-k})\right)\right\}, \qquad (1.11)$$
$$u_{N-k} \in U_{N-k}(x_{N-k-1}), \qquad f_{N-k}(x_{N-k-1}, u_{N-k}) \in Y_k,$$

де в обмеженнях остання умова забезпечує потрапляння точки x_{N-k} в область визначення функції $S_k(x_{N-k})$.

Нехай $u_{N-k}^*(x_{N-k-1})$ – значення управління, при якому досягається мінімум витрат в (1.11). Вираз $u_k^*(x_{k-1})$ – визначають для кожного моменту

часу оптимальні правила управління у вигляді функцій від поточного стану динамічного процесу, тобто задають закон оптимального управління в формі оптимального регулятора за станом. Оптимальний початковий стан x_0^* можна отримати з вирішення наступного завдання:

$$x_0^* = \arg\min\left\{S_N(x_0) : x_0 \in X_0\right\}. \qquad (1.12)$$

Систему (1.10)-(1.12) називають рекурентними рівняннями Беллмана. Значення оптимального управління в явному вигляді можна послідовно визначити наступним чином:

$$u_1^* = u_1^*(x_0), x_1^* = f_1(x_0^*, u_1^*), ..., u_k^* = u_k^*(x_{k-1}), x_k^* = f_k(x_{k-1}^*, u_k^*) \qquad (1.13)$$

Зауважимо, що рівняння (1.10), (1.11) визначають наступне правило побудови управління: незалежно від того, яким чином керований процес на етапі k потрапив в стан , далі треба застосовувати управління, оптимальне для цього стану в завершальному $(N-k)$ - кроковому процесі з урахуванням оптимального продовження, і в стані x_k потрібно застосовувати правило $u_{k+1}^*(x_k)$, яке визначає перший «такт» такого управління. Це одна з можливих формулювань принципу Беллмана в формі достатньої умови. Такий вибір управління для поточного кроку враховує його вплив на майбутнє, а також те, що наслідки неправильного вибору, зробленого в даний момент, в майбутньому виправити не можна.

Рівняння (1.10) - (1.12) записані у формі, що визначає рішення від кінця процесу. Можливий запис аналогічних рівнянь щодо початку процесу. При цьому функції Беллмана (позначимо їх для нової форми запису через $Z_k(x_k)$) повинні визначати оптимальні витрати при переході з станів початкового безлічі X_0 в стані x_k за k кроків.

Вище вже було зазначено, що при використанні методу Беллмана першим важливим етапом рішення є така постановка вихідної задачі, при якій вона буде укладатися в схему динамічного програмування. Часто це можна зробити декількома різними способами. Важливим є також те, що на основі

рекурентних співвідношень Беллмана можна вирішувати завдання, що трохи відрізняються від наведеної стандартної постановки. Наприклад, функція загальних витрат може не бути адитивною, а мати вигляд твори витрат $d_k\left(x_{k-1},u_k\right)$ по кроках (при позитивності цих витрат) або максимуму з цих витрат. Природно, це призведе до відповідних змін в записи (1.11) рівнянь Беллмана.

2 Постановка задачі. Математичні моделі

2.1 Постановка задачі

Компанія Google останніми роками увійшла в наше життя досить щільно. Сервіси Google свідомо, чи ні, використовує майже кожна людина працездатного віку. Попит на такі сервіси лише зростає. Сучасна молодь, треба визнати, все більше і більше використовує електронні ресурси для навчання, і, попит на паперові носії (книги) знижується. В той же час, треба визнати, якісного контенту ще треба пошукати.

Задля збереження та більш широкого розповсюдження видань світової літератури, як художньої так і наукової 15 років тому був заснований проект повнотекстового пошуку інформації Google Books. З 2005 року Google Books зібрав цифрові копії більш ніж 40 мільйонів книг 400 мовами, лише скануючи книги з бібліотек та видавництв всього світі. Та це лише невеличка частина, адже більшість книг так і залишаються доступними лише у друкованому варіанті, що в сучасному світі є просто недопустимо. У 2011 році цей проект мав певні проблеми юридичного характеру, він дещо змінився, але сканування продовжується. Кількість книг для сканування співробітниками компанії у 2010 році оцінювалось у 130 мільйонів книг. Тому у цій роботі розглянуто проблеми, пов’язані зі скануванням величезної кількості книг, що зберігаються в бібліотеках по всьому світу.

Розглядається задача в такій постановці:

За визначений проміжок часу (T) відсканувати максимальну за вартістю кількість книг, враховуючи такі обмеження:

- перш ніж сканувати з визначеної бібліотеки книги необхідно провести реєстрацію бібліотеки, процес реєстрації для бібліотеки з номером k займає визначений час $T_k^{(r)}$;
- одночасно можлива реєстрація лише однієї бібліотеки, процес сканування не починається до завершення реєстрації;

- різні бібліотеки мають різні можливості сканування, що визначається показником N_k - кількість книг, які можна відсканувати протягом одного дня з бібліотеки k
- книги мають різну вартість C_n;
- кожну книгу можна сканувати лише один раз.

Таким чином, маємо таку оптимізаційну задачу:

$$S = \sum_{i=1}^{n} C_i x_i \to \max \qquad (2.1)$$

$$\sum_{i=1}^{k_b} T_i^{(r)} \le T \quad (2.2)$$

$$\left[\frac{n_k}{N_k} \right] + 1 \le T - \sum_{i=1}^{k} T_i^{(r)} \qquad (2.3)$$

$\{x_i\}_{i=1}^{n}$ - множина книжок, що мають бути відскановані протягом відведеного часу (T), $\left[\frac{n_k}{N_k} \right]$ - визначає кількість днів сканування n_k унікальних книг з бібліотеки k.

Побудуємо для такої задачі евристичні алгоритми і порівняємо їх у застосуванні.

2.2 Алгоритм, що ґрунтується на методі динамічного програмування

Алгоритм ґрунтується на принципі оптимальності, що можна сформулювати так:

Якщо керування оптимальне, то яким би не був початковий стан системи та керування системою в початковий момент часу, наступне керування оптимальне відносно стану, в якому система опиниться в результаті початкового керування

або

Оптимальне керування у будь який момент часу не залежить від попереднього керування (передісторії) і визначається лише станом системи в даний момент часу і метою керування.

Розглянемо нашу задачу, як керовану динамічну систему, стан, якої у кожен момент часу t_k описується вектором $x_k \in R^{n_k}$ (унікальні книги, що вже відскановані до цього моменту часу), а закон зміни її стану визначається співвідношенням: $x_{k+1} = f_{k+1}(x_k, u_{k+1})$. Керування u_{k+1} визначає номер бібліотеки $k+1$, книги з якої слід додати в момент часу t_{k+1} так, щоб критерій якості досягав свого найбільшого значення. Оскільки в завданні маємо обмежений час, то в якості модифікованого критерію якості до (2.1) врахуємо ще й час, що витрачається на реєстрацію та сканування книг

$$\sum_{k=1}^{k_b} \frac{\sum_{i=1}^{n_k} C_i x_i^{(k)}}{\left(T_k^{(r)} + \left[\frac{n_k}{N_k}\right] + 1\right)} \left(\left[\frac{n_k}{N_k}\right] + 1\right) \to \max$$

де множник $\dfrac{\sum_{i=1}^{n_k} C_i x_i^{(k)}}{\left(T_k^{(r)} + \left[\dfrac{n_k}{N_k}\right] + 1\right)}$ визначає швидкість з якою зростає критерій якості в системі, тобто якісний показник для кожної системи.

Отже алгоритм за цим методом можна побудувати так:

Крок 1

Визначаємо кінцеве керування u_{k_b}, тобто обираємо бібліотеку k_b так щоб швидкість зростання сумарної вартості книг бібліотеки має бути максимальною

$$\frac{\sum_{i=1}^{n_{k_b}} C_i x_i^{(k_b)}}{\left(T_{k_b}^{(r)} + \left[\dfrac{n_{k_b}}{N_{k_b}}\right] + 1\right)} \to \max$$

Відповідно, обираємо таку бібліотеку і до вектору стану включаємо книги $x_i^{(k_b)}$ з цієї бібліотеки, тоді

$$S_{k_b} = \sum_{i=1}^{n_{k_b}} C_i x_i^{(k_b)}.$$

Крок 2

Наступну бібліотеку $k_b - 1$ обираємо з умови

$$\frac{\sum_{i=1}^{n_{k_b-1}} C_i x_i^{(k_b-1)}}{\left(T_{k_b-1}^{(r)} + \left[\frac{n_{k_b-1}}{N_{k_b-1}} \right] + 1 \right)} \to \max$$

де число книг, що може бути сканована з відповідної бібліотеки визначається або максимальною кількістю унікальних книг Q_{k_b-1} (після виключення вже від сканованих), або такою кількістю книг, що може бути від сканована на відрізку часу, що розглядається, тобто

$$\left[\frac{n_{k_b-1}}{N_{k_b-1}} \right] + 1 \le T_{k_b}^{(r)} + \left[\frac{n_{k_b}}{N_{k_b}} \right] + 1 \Rightarrow n_{k_b-1} = \min\left(Q_{k_b-1}, \left(T_{k_b}^{(r)} + \left[\frac{n_{k_b}}{N_{k_b}} \right] + 1 \right) N_{k_b-1} \right)$$

Відповідно, обираємо таку бібліотеку і до вектору стану включаємо n_{k_b-1} книг $x_i^{(k_b-1)}$ з цієї бібліотеки, тоді

$$S_{k_b-1} = S_{k_b} + \sum_{i=1}^{n_{k_b-1}} C_i x_i^{(k_b-1)}.$$

Крок 3

Наступні бібліотеки $k_b - m$ обираємо аналогічно з умови

$$\frac{\sum_{i=1}^{n_{k_b-m}} C_i x_i^{(k_b-m)}}{\left(\sum_{i=0}^{m-1} T_{k_b-i}^{(r)} + \left[\frac{n_{k_b-m}}{N_{k_b-m}} \right] + 1 \right)} \to \max$$

Для бібліотек, для яких виконується умова

$$T_{k_b-m}^{(r)} \ge T - \left(\sum_{i=0}^{m-1} T_{k_b-i}^{(r)} + \left[\frac{n_{k_b}}{N_{k_b}} \right] + 1 \right)$$

де число книг n_{k_b-m}, що може бути скановано з відповідної бібліотеки визначається або максимальною кількістю унікальних книг Q_{k_b-m} (після виключення вже від сканованих), або такою кількістю книг, що може бути відсканована на відрізку часу, що розглядається, тобто

$$\left[\frac{n_{k_b-m}}{N_{k_b-m}}\right]+1\le\sum_{i=0}^{m-1}T_{k_b-i}^{(r)}+\left[\frac{n_{k_b}}{N_{k_b}}\right]+1\Rightarrow n_{k_b-m}=\min\left(Q_{k_b-m},\left(\sum_{i=0}^{m-1}T_{k_b-i}^{(r)}+\left[\frac{n_{k_b}}{N_{k_b}}\right]+1\right)N_{k_b-m}\right)$$

Відповідно, обираємо таку бібліотеку і до вектору стану включаємо n_{k_b-m} книг $x_i^{(k_b-m)}$ з цієї бібліотеки, тоді

$$S_{k_b-m}=\sum_{i=0}^{m-1}S_{k_b-i}+\sum_{i=1}^{n_{k_b-m}}C_i x_i^{(k_b-m)}.$$

Цей етап повторюється доти поки не досягнемо ситуації, що реєструвати жодну бібліотеку немає можливості

$$T_{k_b-m_0}^{(r)}>T-\left(\sum_{i=0}^{m_0-1}T_{k_b-i}^{(r)}+\left[\frac{n_{k_b}}{N_{k_b}}\right]+1\right)$$

Крок 4 (уточнення)

До загальної суми тепер можна додати для зареєстрованих бібліотек ті книжки, що в кожній з них можна відсканувати за час

$$T-\left(\sum_{i=0}^{m_0-1}T_{k_b-i}^{(r)}+\left[\frac{n_{k_b}}{N_{k_b}}\right]+1\right)$$

Якщо такі є, то відповідні книги додаємо до вектору стану.

Крок 5

Обираємо новий інтервал часу у вигляді

$$T-\sum_{i=0}^{m_0-1}T_{k_b-i}^{(r)}$$

Тобто повторюємо кроки 1-4 де для часу T слід визначити за формулою

$$T-\sum_{i=0}^{m_0-1}T_{k_b-i}^{(r)}.$$

Повтор проводимо поки для нового часу T сумарна вартість книг може бути збільшена.

У якості висновків до цього методу, можна сформулювати таке:

- Цей метод буде ефективним, якщо час, що відведено на сканування більший за час відведений на сканування обраних книг з бібліотек, що включаються до процесу.
- Для підвищення ефективності спочатку провести аналіз вартісних характеристик книг, щодо розподілу у різних цінових діапазонах (поділивши на відповідні квантилі усі книги).
- Обмежений час суттєво впливає на вибір методу, тому покращити результат можна за рахунок розгляду не повної множини, а певних підмножин, що містять найцінніші видання.

2.3 Алгоритм, що ґрунтується на сортуванні

Суть алгоритму зводиться до сортування бібліотек за певними чинниками та алгоритмами:

1. Визначити максимальну кількість книг, які можна відсканувати в кожній бібліотеці, за кількість днів, що зазначена в умові зменшену на час реєстрації відповідної бібліотеки.

$$n_k = \min\left(Q_k, N_k\left(T - T_k^{(r)}\right)\right), \qquad T - T_k^{(r)} > 0$$

де Q_k - максимальна кількість книг у бібліотеці k; N_k - кількість книг, що можна відсканувати протягом одного дня з бібліотеки k;

2. Серед усіх книг для кожної бібліотеки обрати ті, що мають найбільшу цінність та знайти їх суму, відповідно до пункту 1.

$$S_k = \sum_{i=1}^{n_k} C_i x_i^{(k)}.$$

3. Обрати бібліотеку з найбільшою сумою цінностей $\max S_k$.

4. Виключити з числа книг,ті що відібрані у бібліотеці відібраній у пп.3.

5. Зменшити час для сканування на час реєстрації відібраної бібліотеки $T = T - T_k^{(r)}$.

6. Якщо час T, що лишився, більший за найменший час реєстрації, то повторити пункти 1-3 для бібліотек та книг, що лишились.

7. Знайти загальну суму цінностей книг.

До переваг цього методу можна віднести його

- відносну простоту,
- алгоритмічність.

Але метод має недоліки, що можливо усунути, вивчивши попередньо структури даних. Так, метод суттєво може залежати від обраної оцінки та довжини часового проміжку, що виділено. До недоліків можна віднести

- отриманий розв'язок лише наближено дає результат і може не враховувати, наприклад, маленькі бібліотеки, що містять лише невелику кількість книг, досить високої цінності;
- також за таким алгоритмом використання виділеного часу може бути неефективним, оскільки, в залежності від довжини часового проміжку, сканування деякої бібліотеки може бути завершено до завершення виділеного часу, що при певній оптимізації завершеного алгоритму, можна покращити результат.

Для усунення наведених недоліків можна виконати таке:

- замість оцінки $\max S_k$ на кроці 3 можна обрати дещо іншу оцінку, зокрема, наприклад, швидкість збільшення сумарної цінності книг;
- після попереднього вибору бібліотек, спробувати переставити порядок реєстрації бібліотек, що скануються за час менший, ніж відведено, тим самим збільшити час на сканування інших бібліотек.

2.4 Програмна реалізація алгоритму

Для початку розглянемо набір вхідних даних. Його подано у вигляді простого тексту. Файл містить лише ASCII символи з рядками, що

закінчуються одним символом '\n' . Коли в одному рядку подано кілька чисел, вони відокремлюються одним пробілом між кожними двома числами.

Перший рядок набору даних містить:

- кількість різних книг;
- кількість бібліотек;
- кількість днів для сканування.

Далі йде один рядок, що містить цілі числа, описуючи оцінку кожної книги (рисунок 2.1).

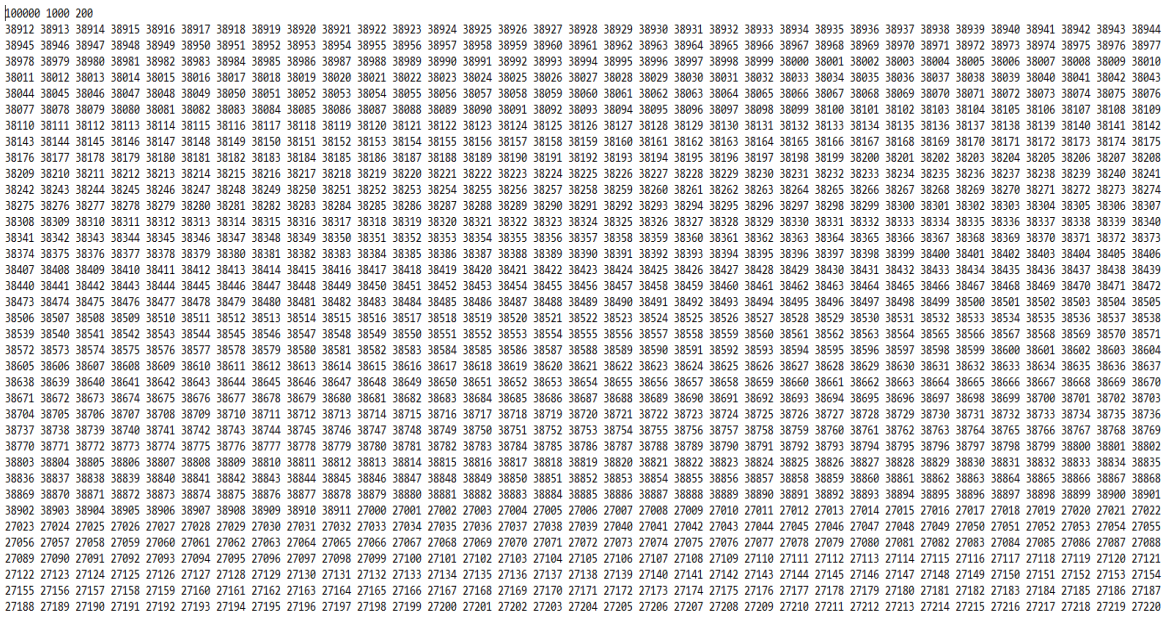

Рис. 2.1 – Оцінка всіх книг

Потім слідують розділи, що описують окремі бібліотеки.

Кожен такий розділ містить два рядки.

Перший рядок містить:

- кількість книг у бібліотеці;
- кількість днів, необхідних для закінчення реєстрації кожної бібліотеки;
- кількість книг, які можна відсканувати протягом одного дня, як тільки бібліотека пройде реєстрацію.

Другий рядок містить цілі числа, що описують ідентифікатори книг у бібліотеці. Кожен ідентифікатор книги вказаний щонайменше один раз на бібліотеку (рисунок 2.2).

```
344 1 2
31074 53941 54810 20886 93442 43074 80092 64691 50618 43297 4213 53602 92380 62989 38183 59126 83261 67463 13782 74214 56984 84637 47044 54664 34352 22580 87833 63263 80650 56286 93817 50592 53706
12421 50872 65211 30475 11006 91865 97566 3951 20380 84835 84948 30407 2885 94267 52440 65959 85809 4113 593 45717 86413 30769 72715 26528 35319 67732 37903 25170 41469 50523 71630 29259 27863
42234 67516 6093 92397 27656 6973 75458 1177 72752 30623 62436 18877 75179 386 59703 77895 82252 80909 25685 80354 14100 95567 35983 38182 12286 59278 20438 31470 34608 32366 92730 3342 11007 63100
34373 98726 60944 40455 89917 61524 22260 3066 48814 42729 9069 4374 74784 29026 42557 36651 7481 23922 4328 50317 76166 44902 80160 5610 24867 35661 18088 1734 39198 52906 30262 44402 2797 60979
77738 99630 24972 26092 8602 89805 54534 85982 33315 5325 86242 18709 2846 86920 85867 45680 91345 8024 19369 79038 42800 47842 37116 37367 85336 35107 52583 7292 84991 28809 73947 99650 55326 7180
8930 14769 81264 45358 89603 10108 73467 52090 73775 8674 8351 42332 39752 7563 9591 1970 13975 50625 91648 21671 23954 2500 32267 96613 42434 76197 58955 48946 9027 92963 22510 70371 83315 58636
12749 93078 35197 73153 83655 46257 38069 35512 41289 79315 31633 40929 45410 12261 78361 39492 82801 32237 11917 74985 52050 90797 17880 70381 85964 62548 71844 42126 40061 80636 51722 23462 48125
71381 96360 43162 9441 19265 28322 77669 97495 54633 55913 37567 86227 15755 37925 693 49247 27998 96897 34505 43658 80191 80118 3313 6600 39973 28604 4290 98633 96284 63214 67555 96727 99567 41060
86342 83834 29628 93968 66827 9857 94965 28683 85368 29655 32003 50047 75985 25592 35444 4476 2407 41071 85925 83832 77763 71846 74650 38494 4731 17643 86636 52465 37940 74370 5683 71126 38880
38677 69055 31819 64075 87332 41328 6580 89497 81433 986 27933 88953 7072 50394 60041 44616 36490 79283 39496 65316 27615 39635 64658 44167 25355 89610 62860 71589 56271 81797 49538 18369 17983
34541 45572 6970 65856 2926 71970 22599 10284 46583
275 8 1
27478 61664 89513 91868 11646 43963 6741 78346 15573 17826 76944 1143 68109 39012 6634 37538 98736 72367 63095 87489 58385 65744 28005 78575 22281 45215 71981 50272 22777 34941 13872 41451 57258
96227 77787 54665 33956 59588 49449 73040 71704 78034 10685 93871 31608 65951 83730 5001 86855 31484 84236 92344 27469 99897 64862 68319 9310 98332 75376 86247 30457 9748 45196 42773 63727 10632
51880 99953 90446 16133 16545 52013 95719 98256 3674 92801 22257 56234 23640 79553 49568 74637 10503 53667 48890 54078 47595 5029 98219 25444 45850 10004 99992 42548 87307 1635 49920 2566 15067
23528 28205 91011 832 73594 60662 78361 63867 7024 42013 12086 30904 52395 24512 80092 7488 2464 73416 3875 48091 34306 88218 8155 67569 19598 25098 16552 99581 23162 80360 6613 75393 20534 76801
42699 87553 56500 58707 65190 93856 9941 23440 88927 60042 57996 83091 38140 48266 59756 18233 10824 13602 52943 43618 60337 24778 75605 31419 33270 41761 19217 91067 29694 26773 23403 57528 27505
23237 35534 90396 15423 62250 74301 90038 5193 85784 6889 13436 86292 85084 19789 74509 32625 1185 34274 13998 90423 38415 10478 68786 30413 69916 89128 14717 83903 79187 52096 32973 38232 71124
94705 91338 42330 75417 62700 55647 35798 35715 41314 60341 23195 56547 10452 48013 69784 51653 13725 13488 66257 24029 59712 15937 87732 73021 78336 21107 16012 55396 92035 69181 50954 3018 19284
95948 73374 15231 24503 87749 86948 57632 64513 40927 88578 57306 52947 57438 7408 61705 67861 16145 82272 35533 60459 42666 61581 55946 49183 97712 98756 87615 70767 73235 37057 1960 96254 23877
31711 35066 76552 62374 21461 53069 76828 79850 75524 98117
42 5 2
14231 20070 21110 31577 28336 7988 46105 73076 4967 9816 75008 18731 15822 80121 59076 28678 33458 40238 80013 94927 94929 15378 32282 990 25309 91892 57618 61729 22850 64189 81648 56751 63065 4797
3694 37448 92753 18725 29076 94531 20417 47713
978 6 1
29482 27966 12927 57777 28522 54497 63843 1250 52245 29278 15514 84059 59735 42755 56440 62917 32337 39245 48764 17578 57479 62796 44971 14792 20623 6828 39075 37364 87731 71152 12032 10361 17271
27929 70219 90472 17825 70134 60495 35847 37365 67221 98022 85717 43065 26257 90117 42762 6839 25031 23310 56061 99447 53309 15784 32960 13785 97979 93853 27763 64947 64978 59966 20397 62094 6541
90820 86353 42606 79039 9366 97171 69520 46154 64337 89820 77129 12232 58003 21140 16739 34165 37573 45834 20859 25972 12255 88037 66839 31746 78909 39257 83803 96556 56357 94129 95421 9783 41646
27020 63776 14703 62891 99106 46298 71812 47458 71404 68784 63505 25132 3657 91891 63540 15927 78365 90944 46022 72316 29354 22926 41651 10759 56508 25909 34539 34777 93092 31766 68503 70257 95085
```

Рис. 2.2 – Оцінка кожної бібліотеки

За основу програми було взято метод, що ґрунтується на сортуванні, який на мою думку є найоптимальнішим. Для початку завантажуємо потрібні бібліотеки (рисунок 2.3).

```
import numpy
from operator import itemgetter
import sys
import pandas
```

Рис. 2.3 – Завантаження стандартних бібліотек

Створюємо клас Library , об’єкти якого повинні мати порядковий номер, загальну кількість книг, кількість днів, яка необхідна на реєстрацію бібліотеки та кількість книг, яку можна відсканувати протягом одного дня (рисунок 2.4).

```
class Library:
    def __init__(self, number, number_of_book, number_of_days_for_registration, books__per_day):
        self.number = number
        self.number_of_book = number_of_book
        self.number_of_days_for_registration = number_of_days_for_registration
        self.books__per_day = books__per_day
```

Рис. 2.4 – Оцінка всіх книг

Задаємо функцію get_best_number_of_book для того, щоб знайти найбільшу кількість книг для кожної бібліотеки, яку можна відсканувати протягом кількості днів заданої (рисунок 2.5).

```
def best_number_of_book(self, first_day_of_scan=0):
    global total_book, l_number_of_days_for_registration, l_books__per_day, days, books_res
    max_days = days - self.number_of_days_for_registration - first_day_of_scan
    actual_b = list(set(self.number_of_book) - set(books_res))
    current_total_book = numpy.take(total_book, actual_b)
    max_books = max(min(int(max_days * self.books__per_day), len(actual_b)), 0)
    if max_books == 0:
        return []
    arg = numpy.argpartition(current_total_book, -max_books)[-max_books:]
    return numpy.take(actual_b, arg)
```

Рис. 2.5 – Функція знаходження максимальної кількості книг

Знаходимо суму найкращих за цінністю книг в кожній бібліотеці, враховуючи їх кількість відповідно до попереднього пункту (рисунок 2.6).

```
def high_result(self, first_day_of_scan=0):
    global total_book, l_number_of_days_for_registration, l_books__per_day, days
    max_days = days - self.number_of_days_for_registration - first_day_of_scan
    max_books = max(min(int(max_days * self.books__per_day), len(self.number_of_book)), 0)
    current_total_book = numpy.take(total_book, self.number_of_book)
    ind = numpy.argpartition(current_total_book, -max_books)[-max_books:]
    high_result = numpy.take(current_total_book, ind)
    return numpy.sum(high_result)
```

Рис. 2.6 – Функція знаходження найцінніших книг

Для сортування бібліотек використовуємо функцію sort_values (рисунок 2.7). Сортування відбувається за спаданням (повний код програми знаходиться в Додатку А).

```
def sort_values(arr_1, arr_2, kind='quicksort', ascending=False):
    if not ascending:
        return zip(*sorted(zip(arr_1, arr_2), key=itemgetter(0))[::-1])
    return zip(*sorted(zip(arr_1, arr_2), key=itemgetter(0)))
```

Рис. 2.7 – Функція сортування

Створюємо функцію sum_total_book для підрахунку загальної суми цінностей книг (рисунок 2.8).

```
def sum_total_book(number_of_book):
    global total_book
    return numpy.sum(numpy.take(total_book, list(number_of_book)))
```

Рис. 2.8 – Функція підрахунку загальної суми цінностей

Створюємо новий файл «result.txt» в якому можна знайти, загальну кількість книг, які були відскановані з якої бібліотеки та їх цінність. Загальний

результат у вигляді суми цінностей усіх книг виводиться на екран (рисунок 2.9).

```
with open(file.replace("books.txt", "result.txt"), "w") as f:
    first_day_of_scan = 0
    for i in range(libr):
        actual_lib = lib_sort[i]
        actual_number_of_book = actual_lib.best_number_of_book(first_day_of_scan)
        books_res.update(actual_number_of_book)
        first_day_of_scan += actual_lib.number_of_days_for_registration
        if len(actual_number_of_book) > 0:
            f.write(str(actual_lib.number) + " " + str(len(actual_number_of_book)) + "\n")
            f.write(str(' '.join(map(str, actual_number_of_book))) + "\n")
        else:
            f.write(str(actual_lib.number) + " 1\n")
            f.write(str(actual_lib.number_of_book[0])+"\n")
    result += sum_total_book(books_res)
    print("Result:", result)
```

Рис. 2.9 – Створення файлу з результатом та вивід результату на екран

Перевірка результату роботи програми.

Для перевірки ефективності алгоритму та правильної роботи програми буде використовуватись один і той же набір даних, але з різної кількістю днів, протягом яких потрібно виконати сканування книг (200 та 300 днів відповідно).

Для першої умови (100 днів для сканування) результат становить 777691526 (рисунок 2.10), а файл «result.txt» виглядає так, як показано на рисунку 2.11.

```
Result: 777691526
```

Рис. 2.10 – Результат роботи програми. Кількість днів для сортування 100.

```
72 198
81184 81199 81229 81675 81516 81476 81494 81500 81955 81791 81248 81461 81284 81677 81858 81489 82031 83413 82141 82759 82490 83348 82268 83116
83625 85584 84186 84023 85032 84804 85355 85744 83730 85802 83680 84387 84021 84808 84903 84811 84695 83665 83990 84849 84648 83647 85137 84869
85938 84213 83930 84724 86109 97706 95105 94565 93061 96606 97973 95928 91031 94290 89790 94307 89465 95527 94184 86125 94182 95066 90966 89424
91777 86128 87674 98285 92274 90228 93926 96374 94525 97560 90232 96324 95021 87403 96322 89046 89065 98039 92974 93163 90234 90174 90417 96539
87625 94617 88348 89858 88249 92292 97858 88477 97318 97550 93757 95287 92639 97047 94491 90902 90389 96533 95529 97027 88833 98106 89915 94968
98489 90095 91970 91225 94024 87148 94950 94629 91996 91495 93720 87158 88210 93538 89948 97803 93313 97415 94908 91454 88756 96408 96934 91645
96128 98178 93328 96777 95559 86469 92321 88723 91543 91620 88103 93664 89417 87459 88321 86657 87227 98469 96468 96890 89277 87244 86265 88069
90719 87494 94664 90688 92735 96831 89509 96427 96488 92400 89335 88628 93609 98505 99011 98744 98673 99170 99174 99250 99931 99384 99414 99473
99474 99800 98560 99588 99601 98798
717 196
79549 80382 80043 79569 80105 80415 80522 81854 82130 81651 81538 81400 81762 81621 82200 82106 81602 81879 81740 81924 80615 81853 81891 82325
82570 82369 82618 82625 82393 82663 86428 82799 89721 87134 83045 90711 89540 89715 84382 88950 88980 85607 87996 85045 85643 87692 89529 87480
84591 88768 83881 88693 84341 82852 90957 90701 87967 86858 90468 84319 84316 86853 86352 86674 84032 88898 88212 86599 83646 86851 83650 86692
86452 87190 87644 90429 85881 86593 89972 86707 88026 83423 85166 88757 90808 89430 82983 87533 90288 87908 90408 83489 89582 89617 85853 89060
82896 89283 83142 89623 89387 87884 89376 88605 86744 84486 85278 88827 84506 83767 87863 88441 91088 96998 92897 96478 92895 94434 95464 99096
94437 91342 97572 99113 91443 95541 98568 97593 92979 99642 96547 92866 93932 93918 97080 94515 96111 95395 97633 93346 93345 94893 95030 91295
95589 94882 94090 95923 98195 93556 98651 94110 96612 97689 94335 91552 96633 99150 99152 99155 97930 98926 98920 94612 92254 99158 91250 92758
95070 93643 97912 96678 97343 99790 93081 97743 97184 99941 96690 98750 99804 92215 98351 93672 92719 95801 91639 95764 97207 92654 96258 94229
91692 97828 92680 99871
972 194
80635 80829 80911 81265 81419 81342 81987 80976 82071 81083 81449 81310 82086 87678 90683 86177 90267 82091 87350 91376 88685 87147 84617 89709
88728 84142 83042 88594 91804 90127 87655 84636 87359 90031 83020 83552 83635 84669 90828 84688 83537 86454 88786 85691 84702 85271 86436 87113
82336 86424 84149 87323 87612 90872 87180 89802 91703 91276 85908 88464 82319 87763 86066 87186 92046 82683 89639 88430 85735 91880 88417 83280
82697 86809 84097 84088 89839 90944 87397 89573 84838 88954 86908 89352 88015 91025 89963 84885 89348 83816 85765 83414 84069 90404 84165 91929
83929 83745 88339 83752 92371 89547 88327 83756 86946 86949 91956 87484 91142 92127 91126 89038 83984 83141 91567 84951 83352 88298 85841 86250
87891 87015 86246 88287 91993 87024 92167 85405 92763 98578 98580 94490 94054 94513 96590 95988 96447 98161 93939 98671 93915 98010 96146 94611
94618 94641 94677 94737 98838 96799 93825 97920 94115 94794 94369 96848 98959 93793 97004 98456 97029 96402 93735 98230 94982 98242 97809 95757
95009 95745 97074 97784 99142 99156 93063 95128 97745 97179 95142 96339 93096 97196 97287 97646 95241 93236 99396 97392 99444 93300 95510 93434
97506 99569
```

Рис. 2.11 –Цінність книг, які були відскановані

Для другої умови (200 днів для сканування) результат становить 2011990408 (рисунок 2.12), а файл «result.txt» виглядає так, як показано на рисунку 2.13.

```
Result: 2011990408
```

Рис. 2.12 – Результат роботи програми. Кількість днів для сортування 200.

```
972 398
78379 78398 78400 36742 36875 36087 36991 36115 36040 36121 51693 52183 20902 52420 39208 20482 36378 20422 20802 52551 20443 36199 20287 52268 51877 20096 20671 20821 20303 39155 20728 20423 36553
39272 39960 11707 11511 39427 11531 11882 11481 11644 11004 11643 39744 39317 39773 39819 11417 11308 11210 8181 85271 40226 87323 29988 66840 29994 95510 40205 89352 89348 87350 29946 93434 99569
91376 87359 97506 83141 5312 30022 81083 60594 60592 60581 5452 68765 83280 87186 60563 99474 40283 91276 60768 87180 87397 60545 78971 40297 97646 81265 78969 93300 68724 99444 97392 87147 29801
79251 83352 85405 81310 40041 29799 83042 29796 30123 80976 83020 91567 87113 68678 99396 93236 66594 30135 68632 78865 80911 87484 79291 95241 81342 97287 89547 91142 5114 97745 29691 83414 92167
60402 87024 87015 84951 89038 79328 66507 80829 97196 89573 93096 95142 86949 40431 86946 60321 97179 95128 84885 97784 91025 5632 95745 93063 81419 79371 95757 86908 29563 97809 69139 88954 84838
60954 29542 60255 78683 89639 93735 99156 81449 99142 60229 30259 90944 91703 97074 87612 95009 86809 69186 60183 66318 82697 83537 29446 78600 94982 97029 82683 83552 80635 93793 79456 30308 90872
97004 87655 66284 80611 84702 89709 88786 84688 80588 90828 84669 79481 4771 8861 84636 87678 88728 97920 98959 84617 5763 93825 80505 88685 16998 30355 67223 92763 96848 16973 61084 94794 91804
69283 16967 90683 29232 61102 8745 16929 83635 96799 98838 12820 94737 88594 80400 85691 61117 66063 8701 8663 94677 65996 80331 69317 86454 16074 69321 89802 61134 94641 86436 16800 82336 87763
86424 81621 98010 93915 94618 16096 59799 85735 91880 94611 5864 5868 88464 82319 8567 89839 93939 95988 98671 88430 88417 80230 4452 61177 65889 80218 4440 8532 12627 85765 96590 16712 94513 4398
91929 90404 83745 94490 83752 98580 88339 98578 83756 12076 12560 91956 16184 88327 16187 12093 65795 30974 30973 59634 59633 67407 88298 85841 86250 87891 16211 86246 88287 91993 12511 16604 16599
65748 92371 69472 84165 94054 83816 89963 65729 96447 98161 12475 8375 92046 84149 96146 85908 65712 84142 79767 80044 69531 82091 82086 98469 94115 86177 83881 94369 90267 79790 90031 30640 98456
82071 96402 12432 98230 61576 98242 4035 65672 4227 84097 8320 84088 61385 4034 88015 8296 84069 83929 79972 96339 8272 92127 81987 30782 79925 86066 12333 61464 90127 12274 79859 83984 30732 91126
522 396
36133 36151 36276 20065 36285 36517 20032 20107 11714 20373 8022 11277 39306 68638 20709 8895 20215 8559 11867 39229 20364 29705 8186 68826 29260 51801 20710 29909 51964 39821 11875 20248 20467
20991 11339 52304 11697 29876 68909 29685 8152 20154 39102 29670 51720 8738 51887 29666 29145 8472 29403 20134 52558 29125 20397 39577 8200 39384 29842 20954 52360 11813 39109 39891 8430 8016 68727
52336 39738 29009 11647 68937 30888 4044 4255 66943 60354 30084 30067 66146 65797 61058 30090 60811 4931 30872 66518 4051 61369 67543 30366 12566 30769 60741 4241 66981 69033 69235 69234 65940
12759 60095 69042 59938 12758 65675 67262 60855 65770 66565 61467 60190 66570 60660 67560 30555 60260 69062 61274 67320 60876 12285 69191 69369 4323 59935 59854 66780 12641 69370 30839 4080 65862
59962 59735 30485 30774 69112 30830 66743 4092 30806 60585 69405 66071 66103 4168 69164 30550 65870 4750 59925 67355 30506 61544 30539 4973 12832 79392 87587 81440 82805 79749 84161 16576 79373
16250 40457 87925 81002 87559 78701 5629 40443 89576 88280 84882 86462 88985 82844 5582 80057 80744 12181 87960 78680 85590 85591 79449 89532 88296 80801 81332 81510 12188 84440 16821 88687 85924
80720 85627 5038 5528 79791 79244 80706 89014 88001 82878 79817 83320 82081 88494 79210 16023 5477 88888 78648 89433 5789 83288 86294 86441 87004 89055 16531 83627 5092 16947 88207 79153 81191
40231 84372 5402 78631 84998 79559 86822 81144 81611 84469 85710 86519 83668 86799 86148 85239 16886 82498 12124 89327 87785 82966 5355 82046 84545 84734 85017 89317 89318 5342 85038 88771 81902
82045 86337 16389 40146 80948 82643 5183 85783 87230 85182 83017 16155 85072 80205 5277 85122 78974 81703 5929 80435 83033 5243 81018 79692 85836 87132 87875 81004 85829 84699 89648 91967 90107
91965 89898 90840 96033 91934 90621 96006 93943 91886 96762 93931 98805 91997 97038 99120 90929 91824 92850 92638 93001 93812 98379 91720 91717 97133 95794 95781 94869 89976 91691 92033 99879 92817
96714 93041 91667 95100 96706 96313 95732 91606 96797 99794 96145 95692 95687 99782 99781 90744 90552 91568 91564 98358 95652 96876 92779 93600 95642 95635 98232 97682 97681 97200 96494 93115 95615
92197 93560 93137 94485 93531 99641 99315 94492 96550 94238 95502 99597 96850 97548 95497 91380 98276 91158 94185 90088 91370 90432 97332 94556 95304 93354 93266 90682 93318 92495 93283 93287
72 394
15162 15211 15303 15472 43659 44493 43911 44488 44367 44409 43733 43961 44313 44049 44191 41485 40143 79060 79066 36059 81122 8455 4360 87244 29887 52412 60658 59651 88321 89277 98560 20727 89335
80137 87227 60659 86265 96533 83116 84213 99588 29957 68774 29959 92400 5286 97550 68773 36000 90389 97560 99473 85137 8425 66844 93328 81184 95527 66856 97415 95529 96488 81199 81025 93313 68925
91454 78973 87158 52548 94491 95559 89417 30025 78959 4319 81229 87148 80994 89424 82141 84186 88348 96539 91225 99414 11348 5200 81248 52574 93538 90417 39124 91495 96468 59698 85355 87403 98505
66629 30066 36216 65852 89465 94525 99384 5505 81284 36229 95287 68655 85032 83348 66965 78010 97318 91543 66585 87459 66584 89509 5134 93609 97706 60845 11265 11252 93163 66998 98489 89065 5082
30142 82268 96606 89046 87494 30152 29640 29639 79307 39879 94565 11204 88249 16743 99250 84903 83413 99800 12464 8561 93664 98673 91620 11749 79334 80241 5610 96427 69101 52122 60916 91031 84869
16548 16767 91645 39075 93061 16547 92321 97803 95105 52091 5648 65922 93720 94617 39793 84849 99174 60963 96408 88477 99170 36390 88210 20623 20895 65952 81461 95066 77963 90966 93757 29520 84387
94629 97858 81476 4233 84811 84808 11848 82759 87625 30284 84804 65968 81489 85584 95021 92292 81494 4912 99931 92974 81500 4909 4899 67169 61028 78626 98744 39706 39040 81516 90234 41081 97047
61051 87674 90232 96374 91777 90228 92274 90902 5771 80655 78607 86128 82031 30357 86125 66307 94307 40609 97027 12386 88833 83625 39677 97973 95928 79546 86109 89790 83647 94968 67268 4544 8951
36944 84724 40656 83665 86469 94290 94950 29406 83680 79589 93926 85744 4554 94664 84695 20561 66263 65994 12751 16454 12024 98039 78541 5878 99011 96324 80577 5887 94908 96322 5892 89858 29370
61189 79626 20234 81675 81677 67342 16144 83730 36635 85802 88756 84648 96934 90174 51676 51872 98106 89915 92639 60056 78487 91970 8853 94024 84023 88723 51678 91996 80520 29320 84021 61494 4742
89948 86657 98798 40806 96890 77689 4711 8060 78435 81791 96128 98178 8070 69511 69510 51809 90719 77867 4137 59996 88103 39515 39502 4131 8111 8770 16305 85938 67516 36797 92735 81858 67528 90688
96831 59966 51773 81955 40915 82490 61398 88628 29188 78385 83930 36827 59945 83990 77837 80420 94182 61414 94184 98285 8736 16901 90095 80390 4106 88069 78355 96777 8719 99601
```

Рис. 2.13 –Цінність відсканованих книг

Як можна бачити, алгоритм працює чудово, а його результат залежить від кількості днів сканування, які задаються в умові.

ВИСНОВКИ

В роботі розроблено математичні моделі задачі про розподіл робіт між бібліотеками по скануванню книг без дублювання за наявності обмежень до ресурсів та часу виконання робіт та реєстрації учасників проекту. В такій постановці завдання є прикладом комбінаторної оптимізації, що можна віднести до узагальненої проблеми про призначення. Загальних детермінованих підходів для розв’язання таких задач немає. Розробка моделей і алгоритмів потребує комбінування різних підходів, що у більшості випадків носять евристичний характер, або потребують використання ітеративних підходів, що ґрунтуються на загальних методах розв’язання задач про призначення (угорський метод та метод Мака).

В рамках магістерської роботи, було розроблено два алгоритми для вирішення поставленої задачі: алгоритм, що ґрунтується на методі динамічного програмування та алгоритм, що ґрунтується на сортуванні і побудовано для них математичні моделі.

Аналіз побудованих алгоритмів дає можливість визначити сильні та слабкі сторони кожного з них. Зокрема, метод, що використовує принцип оптимальності Беллмана, не може вважатись детермінованим і носить евристичний характер, за ним можна зробити такі висновки:

- метод буде ефективним, якщо час, що відведено на сканування більший за час відведений на сканування обраних книг з бібліотек, що включаються до процесу;
- для підвищення ефективності слід попередньо провести аналіз вартісних характеристик книг, щодо розподілу у різних цінових діапазонах (поділивши на відповідні квантилі усі книги);
- обмежений час суттєво впливає на вибір методу, тому покращити результат можна за рахунок розгляду не повної множини, а певних підмножин, що містять найцінніші видання.

До переваг методу, що використовує ітераційнє сортування множин можна віднести його

- відносну простоту,
- алгоритмічність.

Але метод має недоліки, що можливо усунути, вивчивши попередньо структури даних. Так, метод суттєво може залежати від обраної оцінки та довжини часового проміжку, що виділено. До недоліків можна віднести:

- отриманий розв’язок лише наближено дає результат і може не враховувати, наприклад, маленькі бібліотеки, що містять лише невелику кількість книг, досить високої цінності;
- за таким алгоритмом використання виділеного часу може бути неефективним, оскільки, в залежності від довжини часового проміжку, сканування деякої бібліотеки може бути завершено до завершення виділеного часу, що при певній оптимізації завершеного алгоритму, можна покращити результат.

Для усунення наведених недоліків можна виконати таке:

- оптимізувати метод за рахунок підбору найбільш вдалого критерію якості, або комбінації різних критеріїв;
- поліпшити отриманий результат за рахунок перестановки порядку реєстрації бібліотек з метою збільшення критерію якості моделі.

В ході роботи було створено програму за останнім алгоритмом.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Assignment Problem in Linear Programming : Introduction and Assignment Model [Electronic resource] . – Mode of access : https://www.yourarticlelibrary.com/ergonomics/operation-research/assignment-problem-in-linear-programming-introduction-and-assignment-model/34712

2. Munapo E., Development of an accelerating hungarian method for assignment problems ,2020. – 13 с.

3. Алгоритм решения задачи о назначениях [Електронний ресурс]. – Режим доступу : http://prepod2000.kulichki.net/item_220.html

4. Гольштейн Е., Юдин Д. Задачи и методы линейного программирования. Математические основы и практические задачи, 2010. – 322 с.

5. Лелякова Л. В., Харитонова А. Г., Чернишова Г. Д.; ВЕСТНИК ВГУ " Воронежский государственный университет ", 2017. – 27 с.

6. Решение минимаксной распределительной задачи методом динамического програмирования с применением паралельных вычислений [Електронний ресурс]. – Режим доступу : http://agora.guru.ru/abrau2011/pdf/580.pdf

7. Brommess P., «Solving the Generalized Assignment Problem by column enumeration based on Lagrangian reduced costs», 2005. – 41 с.

8. Венгерский метод решения задач о назначениях [Електронний ресурс]. – Режим доступу : https://studfile.net/preview/1467094/#3

9. Венгерский алгоритм [Електронний ресурс]. – Режим доступу : – https://e-maxx.ru/algo/assignment_hungary

10. Using Interval Operations in the Hungarian Method to Solve the Fuzzy Assignment Problem and Its Application in the Rehabilitation Problem of Valuable Buildings in Egypt [Electronic resource] . – Mode of access : https://www.hindawi.com/journals/complexity/2020/9207650/

11. Hungarian method for solving assignment problem - quantitative

techniques for management [Electronic resource] . – Mode of access : https://www.wisdomjobs.com/e-university/quantitative-techniques-for-management-tutorial-297/hungarian-method-for-solving-assignment-problem-9898.html

12. Assignment Problem and Hungarian Algorithm [Electronic resource] . – Mode of access : https://www.topcoder.com/community/competitive-programming/tutorials/assignment-problem-and-hungarian-algorithm/

13. Банди Б. «Основы линейного программирования», 1989. – 174 с.

14. Введение в исследование операций [Електронний ресурс]. – Режим доступу : http://edu.tsu.ru/eor/resourse/578/tpl/index.html

15. Неймарк Ю.И. Динамические системы и управляемые процессы. – М.:Наука, 1978.

16. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования. – М.: Наука, 1965.

17. Калихман И.Л., Войтенко М.А. Динамическое программирование в примерах и задачах. – М.: Высшая школа, 1979.

18. Арис Р. Дискретное динамическое программирование. – М.: Мир, 1969.

19. Стронгин Р.Г. Численные методы в многоэкстремальных задачах(информационно-статистические алгоритмы). – М.: Наука, 1978.

Додатки

Додаток А

books.txt

100000 1000 300

38912 38913 38914 38915 38916 38917 38918 38919 38920 38921 38922 38923
38924 38925 38926 38927 38928 38929 38930 38931 38932 38933 38934 38935
38936 38937 38938 38939 38940 38941 38942 38943 38944 38945 38946 38947
38948 38949 38950 38951 38952 38953 38954 38955 38956 38957 38958 38959
38960 38961 38962 38963 38964 38965 38966 38967 38968 38969 38970 38971
38972 38973 38974 38975 38976 38977 38978 38979 38980 38981 38982 38983
38984 38985 38986 38987 38988 38989 38990 38991 38992 38993 38994 38995
38996 38997 38998 38999 38000 38001 38002 38003 38004 38005 38006 38007
38008 38009 38010 38011 38012 38013 38014 38015 38016 38017 38018 38019
38020 38021 38022 38023 38024 38025 38026 38027 38028 38029 38030 38031
38032 38033 38034 38035 38036 38037 38038 38039 38040 38041 38042 38043
38044 38045 38046 38047 38048 38049 38050 38051 38052 38053 38054 38055
38056 38057 38058 38059 38060 38061 38062 38063 38064 38065 38066 38067
38068 38069 38070 38071 38072 38073 38074 38075 38076 38077 38078 38079
38080 38081 38082 38083 38084 38085 38086 38087 38088 38089 38090 38091
38092 38093 38094 38095 38096 38097 38098 38099 38100 38101 38102 38103
38104 38105 38106 38107 38108 38109 38110 38111 38112 38113 38114 38115
38116 38117 38118 38119 38120 38121 38122 38123 38124 38125 38126 38127
38128 38129 38130 38131 38132 38133 38134 38135 38136 38137 38138 38139
38140 38141 38142 38143 38144 38145 38146 38147 38148 38149 38150 38151
38152 38153 38154 38155 38156 38157 38158 38159 38160 38161 38162 38163
38164 38165 38166 38167 38168 38169 38170 38171 38172 38173 29439 29440
29441 29442 29443 29444 29445 29446 29447 29448 29449 29450 29451 29452
29453 29454 29455 29456 29457 29458 29459 29460 29461 29462 29463 29464

29465 29466 29467 29468 29469 29470 29471 29472 29473 29474 29475 29476
29477 29478 29479 29480 29481 29482 29483 29484 29485 29486 29487 29488
29489 29490 29491 29492 29493 29494 29495 29496 29497 29498 29499 29500
29501 29502 29503 29504 29505 29506 29507 29508 29509 29510 29511 29512
29513 29514 29515 29516 29517 29518 29519 29520 29521 29522 29523 29524
29525 29526 29527 29528 29529 29530 29531 29532 29533 29534 29535 29536
29537 29538 29539 29540 29541 29542 29543 29544 29545 29546 29547 29548
29549 29550 29551 29552 29553 29554 29555 29556 29557 29558 29559 29560
29561 29562 29563 29564 29565 29566 29567 29568 29569 29570 29571 29572
29573 29574 29575 29576 29577 29578 29579 29580 29581 29582 29583 29584
29585 29586 29587 29588 29589 29590 29591 29592 29593 29594 29595 29596
29597 29598 29599 29600 29601 29602 29603 29604 29605 29606 29607 29608
29609 29610 29611 29612 29613 29614 29615 29616 29617 29618 29619 29620
29621 29622 29623 29624 29625 29626 29627 29628 29629 29630 29631 29632
29633 29634 29635 29636 29637 29638 29639 29640 29641 29642 29643 29644
29645 29646 29647 29648 29649 29650 29651 29652 29653 29654 29655 29656
29657 29658 29659 29660 29661 29662 29663 29664 29665 29666 29667 29668
29669 29670 29671 29672 29673 29674 29675 29676 29677 29678 29679 29680
29681 29682 29683 29684 29685 29686 29687 29688 29689 29690 29691 29692
29693 29694 29695 29696 29697 29698 29699 29700 29701 29702 29703 29704
29705 29706 29707 29708 29709 29710 29711 29712 29713 29714 29715 29716
29717 29718 29719 29720 29721 29722 29723 29724 29725 29726 29727 29728
29729 29730 29731 29732 29733 29734 29735 29736 29737 29738 29739 29740
29741 29742 29743 29744 29745 29746 29747 29748 29749 29750 29751 29752
29753 29754 29755 29756 29757 29758 29759 29760 29761 29762 29763 29764
29765 29766 29767 29768 29769 29770 29771 29772 29773 29774 29775 29776
29777 29778 29779 29780 29781 29782 29783 29784 29785 29786 29787 29788
29789 29790 29791 29792 29793 29794 29795 29796 29797 29798 29799 29800
29801 29802 29803 29804 29805 29806 29807 29808 29809 29810 29811 29812
29813 29814 29815 29816 29817 29818 29819 29820 29821 29822 29823 29824

29825 29826 29827 29828 29829 29830 29831 29832 29833 29834 29835 29836
29837 29838 29839 29840 29841 29842 29843 29844 29845 29846 29847 29848
29849 29850 29851 29852 29853 29854 29855 29856 29857 29858 29859 29860
29861 29862 29863 29864 29865 29866 29867 29868 29869 29870 29871 29872
29873 29874 29875 29876 29877 29878 29879 29880 29881 29882 29883 29884
29885 29886 29887 29888 29889 29890 29891 29892 29893 29894 29895 29896
29897 29898 29899 29900 29901 29902 29903 29904 29905 29906 29907 29908
29909 29910 29911 29912 29913 29914 29915 29916 29917 29918 29919 29920
29921 29922 29923 29924 29925 29926 29927 29928 29929 29930 29931 29932
29933 29934 29935 29936 29937 29938 29939 29940 29941 29942 29943 29944
29945 29946 29947 29948 29949 29950 29951 29952 29953 29954 29955 29956
29957 29958 29959 29960 29961 29962 29963 29964 29965 29966 29967 29968
29969 29970 29971 29972 29973 29974 29975 29976 29977 29978 29979 29980
29981 29982 29983 29984 29985 29986 29987 29988 29989 29990 29991 29992
29993 29994 29995 29996 29997 29998 29999 48000 48001 48002 48003 48004
48005 48006 48007 48008 48009 48010 48011 48012 48013 48014 48015 48016
48017 48018 48019 48020 48021 48022 48023 48024 48025 48026 48027 48028
48029 48030 48031 48032 48033 48034 48035 48036 48037 48038 48039 48040
48041 48042 48043 48044 48045 48046 48047 48048 48049 48050 48051 48052
48053 48054 48055 48056 48057 48058 48059 48060 48061 48062 48063 48064
48065 48066 48067 48068 48069 48070 48071 48072 48073 48074 48075 48076
48077 48078 48079 48080 48081 48082 48083 48084 48085 48086 48087 48088
48089 48090 48091 48092 48093 48094 48095 48096 48097 48098 48099 48100
48101 48102 48103 48104 48105 48106 48107 48108 48109 48110 48111 48112
48113 48114 48115 48116 48117 48118 48119 48120 48121 48122 48123 48124
48125 48126 48127 48128 48129 48130 48131 48132 48133 48134 48135 48136
48137 48138 48139 48140 48141 48142 48143 48144 48145 48146 48147 48148
48149 48150 48151 48152 48153 48154 48155 48156 48157 48158 48159 48160
48161 48162 48163 48164 48165 48166 48167 48168 48169 48170 48171 48172
48173 48174 48175 48176 48177 48178 48179 48180 48181 48182 48183 48184

48185 48186 48187 48188 48189 48190 48191 48192 48193 48194 48195 48196
48197 48198 48199 48200 48201 48202 48203 48204 48205 48206 48207 48208
48209 48210 48211 48212 48213 48214 48215 48216 48217 48218 48219 48220
48221 48222 48223 48224 48225 48226 48227 48228 48229 48230 48231 48232
48233 48234 48235 48236 48237 48238 48239 48240 48241 48242 48243 48244
48245 48246 48247 48248 48249 48250 48251 48252 48253 48254 48255 48256
48257 48258 48259 48260 48261 48262 48263 48264 48265 48266 48267 48268
48269 48270 48271 48272 48273 48274 48275 48276 48277 48278 48279 48280
48281 48282 48283 48284 48285 48286 48287 48288 48289 48290 48291 48292
48293 48294 48295 48296 48297 48298 48299 48300 48301 48302 48303 48304
48305 48306 48307 48308 48309 48310 48311 48312 48313 48314 48315 48316
48317 48318 48319 48320 48321 48322 48323 48324 48325 48326 48327 48328
48329 48330 48331 48332 48333 48334 48335 48336 48337 48338 48339 48340
48341 48342 48343 48344 48345 48346 48347 48348 48349 48350 48351 48352
48353 48354 48355 48356 48357 48358 48359 48360 48361 48362 48363 48364
48365 48366 48367 48368 48369 48370 48371 48372 48373 48374 48375 48376
48377 48378 48379 48380 48381 48382 48383 48384 48385 48386 48387 48388
48389 48390 48391 48392 48393 48394 48395 48396 48397 48398 48399 48400
48401 48402 48403 48404 48405 48406 48407 48408 48409 48410 48411 48412
48413 48414 48415 48416 48417 48418 48419 48420 48421 48422 48423 48424
48425 48426 48427 48428 48429 48430 48431 48432 48433 48434 48435 48436
48437 48438 48439 48440 48441 48442 48443 48444 48445 48446 48447 48448
48449 48450 48451 48452 48453 48454 48455 48456 48457 48458 48459 48460
48461 48462 48463 48464 48465 48466 48467 48468 48469 48470 48471 48472
48473 48474 48475 48476 48477 48478 48479 48480 48481 48482 48483 48484
48485 48486 48487 48488 48489 48490 48491 48492 48493 48494 48495 48496
48497 48498 48499 48500 48501 48502 48503 48504 48505 48506 48507 48508
48509 48510 48511 48512 48513 48514 48515 48516 48517 48518 48519 48520
48521 48522 48523 48524 48525 48526 48527 48528 48529 48530 48531 48532
48533 48534 48535 48536 48537 48538 48539 48540 48541 48542 48543 48544

48545 48546 48547 48548 48549 48550 48551 48552 48553 48554 48555 48556
48557 48558 48559 48560 48561 48562 48563 48564 48565 48566 48567 48568
48569 48570 48571 48572 48573 48574 48575 48576 48577 48578 48579 48580
48581 48582 48583 48584 48585 48586 48587 48588 48589 48590 48591 48592
48593 48594 48595 48596 48597 48598 48599 48600 48601 48602 48603 48604
48605 48606 48607 48608 48609 48610 48611 48612 48613 48614 48615 48616
48617 48618 48619 48620 48621 48622 48623 48624 48625 48626 48627 48628
48629 48630 48631 48632 48633 48634 48635 48636 48637 48638 48639 48640
48641 48642 48643 48644 48645 48646 48647 48648 48649 48650 48651 48652
48653 48654 48655 48656 48657 48658 48659 48660 48661 48662 48663 48664
48665 48666 48667 48668 48669 48670 48671 48672 48673 48674 48675 48676
48677 48678 48679 48680 48681 48682 48683 48684 48685 48686 48687 48688
48689 48690 48691 48692 48693 48694 48695 48696 48697 48698 48699 48700
48701 48702 48703 48704 48705 48706 48707 48708 48709 48710 48711 48712
48713 48714 48715 48716 48717 48718 48719 48720 48721 48722 48723 48724
48725 48726 48727 48728 48729 48730 48731 48732 48733 48734 48735 48736
48737 48738 48739 48740 48741 48742 48743 48744 48745 48746 48747 48748
48749 48750 48751 48752 48753 48754 48755 48756 48757 48758 48759 48760
48761 48762 48763 48764 48765 48766 48767 48768 48769 48770 48771 48772
48773 48774 48775 48776 48777 48778 48779 48780 48781 48782 48783 48784
48785 48786 48787 48788 48789 48790 48791 48792 48793 48794 48795 48796
48797 48798 48799 48800 48801 48802 48803 48804 48805 48806 48807 48808
48809 48810 48811 48812 48813 48814 48815 48816 48817 48818 48819 48820
48821 48822 48823 48824 48825 48826 48827 48828 48829 48830 48831 48832
48833 48834 48835 48836 48837 48838 48839

626 9 1

43296 9162 43920 83088 45161 61563 37636 28229 15518 77250 41554 18536
43118 98372 7910 59718 67552 7728 11213 23805 88101 11343 15816 86108
85590 72382 29972 59086 78429 31401 72128 49981 72003 57162 91729 30174
81635 83754 96874 88120 18010 52247 49977 60571 40744 35938 52015 79347

61653 27563 2555 3422 41925 37633 83908 49328 66283 87781 44671 80682 36662 78809 55724 86982 95196 75476 48402 63266 32135 52513 66845 93696 65633 77576 90928 55820 50773 97217 30783 70532 28498 74841 21626 50658 67454 48425 51547 66027 25899 72293 31628 47557 1670 93359 41996 55874 22772 78927 97864 64878 80932 1229 88113 45190 68695 57042 64263 71492 38162 54693 96393 33798 86926 64535 61816 50644 86048 11814 45244 31607 12332 73423 6529 904 1181 80218 2329 13940 8667 52844 26494 33778 6018 39247 18890 98243 50670 54083 67882 25543 66991 94862 30038 32221 60876 1105 38447 42720 77184 85444 1064 33381 62565 55448 96682 95262 7352 7316 32454 92910 27603 19913 42866 19109 96950 55863 66441 82751 29136 18818 52362 35546 70841 42354 58183 55418 65453 18993 60509 55729 7762 52289 91466 24726 17086 48566 33110 55912 32420 36426 76224 82005 44441 49881 96059 90189 18800 83775 92579 6285 55200 34472 47414 2311 20461 51157 41224 4180 6210 77070 25068 32997 85492 6413 43128 16476 86220 31254 4531 57561 77140 6765 50269 25279 96526 98808 88601 36101 8333 50656 46278 72601 93285 1722 44350 78134 26686 56464 63933 27167 66692 54552 67173 57296 77555 84547 49135 37552 72023 48231 40207 90436 59628 5980 63370 58037 53461 87851 69311 73772 85835 90689 46910 16638 38025 48764 90548 99008 89380 5089 22599 26039 43365 75441 30233 22 74383 52106 90432 17331 34989 77563 87617 8307 35664 44837 79598 20828 22979 97353 33840 79967 22171 22848 59692 8245 49118 87792 80470 86032 90450 72848 90440 84521 30432 62600 85655 4697 98417 61062 92257 51861 68490 48748 17712 95089 58638 88025 97119 9621 6605 78958 96324 34916 33419 58986 83687 54099 90784 46313 61286 42429 98667 51158 35046 6743 39123 91148 21167 71617 94520 45317 97827 57679 55171 58595 84881 27233 80447 23438 66445 27800 68537 71902 18180 12124 94091 77002 45972 50510 73920 57288 61194 98026 29478 75021 76682 28116 61896 97716 80499 92208 71410 91794 61869 1262 59970 77033 71117 18383 21777 43310 81295 20399 34103 11583 59293 42877 9461 61851 22233 91586 34726 56616 54548 94423 88572 70289 28515 11079 70087 26518 13506 16480 5570 56848 20056 4326 1511 62839 39036 83864 2246

46331 50166 11620 40772 10723 12217 29894 97772 10269 96677 76347 79855 95634 69132 88256 73283 75834 4408 21498 26666 71103 84992 78357 92023 48983 60411 64967 50534 51250 75711 88261 95962 54152 68769 41023 76999 13761 15801 65398 40150 41954 68434 57533 14475 2213 11133 73567 22493 28502 93854 94914 20972 12529 14541 15932 76895 51630 76690 44870 21069 24510 84116 95380 89746 20714 98602 69642 10501 2354 70333 36873 82782 22160 35930 99236 50687 42806 99975 72025 42844 39931 76340 34928 559 4051 51107 7012 51073 11032 64553 29632 98578 82243 95194 18096 17931 49658 39526 62994 18754 7011 72939 341 92342 78376 71930 76337 96616 9151 82799 94873 34684 8203 48643 37572 70255 87959 20359 48787 11336 11498 53374 70857 77896 13881 47651 97680 36895 83115 76148 5096 98388 92790 12650 34226 17912 83383 31357 4187 37320 19786 63620 73433 3063 29857 70182 60250 61985 79025 86889 99852 62917 96025 47872 90996 60084 80746 17668 25971 81902 6434 76447 7095 57601 77341 96061 79794 65496 26272 83216 82057 87924 64513 28007 26033 76439 58968 45452 87300 74813 40253 93240 49501 49430 73290 51216 43605 87384 54807 69688 56586 36490 21239 64790 50817 18690 65522 24807 94591 8242 7207 90489 9018 39763 43028 67204 62619 65348 80370 11855 63068 68602 52676

152 4 2

20657 47770 22990 31710 65992 55863 74404 48597 1186 2046 60544 4705 47407 52247 23989 18339 30113 65834 87031 1786 6597 65378 45522 35505 60117 67269 21467 41134 76638 67215 49093 98151 60724 91467 50985 49871 27148 13120 91949 58653 55821 33611 43715 20916 42884 83505 3885 6382 19793 37275 16252 30574 48327 71159 8878 39584 77675 1478 83060 21742 76034 75268 47427 3339 68139 39412 18179 8457 98204 63005 60567 52446 58852 95634 64779 67050 85791 13785 23975 69907 88481 41124 95343 12832 95307 39826 81913 47099 41138 51919 74719 64047 59285 93092 62019 44774 22963 15036 93100 37898 88333 13356 79664 97509 76758 93799 2270 67225 16583 19321 73941 68418 49648 54533 70686 13329 35383 86093 45889 83558 43651 11963 37581 40534 24534 71536 22717 31550 40166 22909 18946 82746 36754

2374 87958 57982 25076 84618 23211 65753 39816 85024 15391 11608 72838 6699 54186 18077 69735 63678 30526 7894

151 6 2

37675 20968 61034 6715 44618 1779 96126 2011 79661 26457 15094 41515 83338 23551 1194 34887 69531 656 40104 29198 96047 90725 15161 6093 52470 39363 77292 40516 98440 46808 79469 61965 75992 75821 87260 49253 67142 71240 26228 2928 95082 72691 46029 88375 4040 19904 64468 8582 76616 59909 80104 37628 15298 10294 75130 86222 18331 15946 30741 63392 76541 74018 64062 73659 50704 76594 43864 91383 95167 87081 4647 92737 74739 27983 51673 87896 29140 6482 94895 80290 23498 48224 61985 95496 71256 67248 951 66282 98969 63219 38448 69554 80190 69213 193 91917 620 92776 98510 94404 16732 93207 91123 1376 97226 93978 55152 46732 44756 4478 53893 46220 83441 6225 27640 62753 66482 81602 39004 99379 79885 6297 60555 34565 3138 50639 50308 49522 39690 20239 95771 67382 43677 25977 96165 32172 70459 66865 3261 24641 38124 42084 4298 74046 93695 72866 43208 36804 35241 31607 59965

177 5 2

83116 43257 51031 69779 6076 95747 75713 84046 44169 50961 8721 98569 92708 10633 78763 22890 57194 4162 17413 96427 90186 14898 88833 70487 82411 10508 65301 42064 87946 64944 65927 13783 17100 44785 94585 87720 55283 72662 62005 1174 38859 45910 35406 64475 64179 53251 71302 9008 21604 20362 91650 70477 79403 85474 3807 143 37663 58103 42603 1031 28966 81748 55032 54069 13083 87748 66795 26713 13708 16394 57020 24796 52486 70137 39818 44847 37331 94075 5030 46302 62547 8287 45708 73365 94573 96913 36434 3975 9026 77964 67375 1484 59464 83955 37998 24634 24933 55103 70199 38183 30541 5306 59934 41586 90881 12076 74303 99245 43301 20482 5662 37851 71131 42886 14635 39469 67793 89463 55053 62422 18950 71178 69350 14890 95330 75412 35769 92617 87978 19680 35921 1460 15515 11373 41164 8012 54313 97109 45390 23857 99355 45372 83288 49427 81916 30052 50295 9179 74594 90557 67164 53886 60184 57438 69509 77971 99019 82815

45588 68722 85236 99920 32167 69928 10806 84790 13569 42318 37191 87326 16266 77522 5909 23312 47408 86625 85233

177 8 1

50519 47195 6170 12205 12348 20292 78483 68078 67530 2643 91976 63028 90992 5726 25024 7346 72766 90759 7546 90552 58103 29592 75555 25425 75321 24046 51275 73961 15101 18818 28286 87125 79840 96340 45637 9555 95082 22443 85852 7991 89996 40356 20473 15301 66829 40292 44114 41300 72388 27627 91897 32028 13120 78539 62340 47940 40402 37197 78527 63696 42130 71773 45947 60393 20487 38566 96696 73147 71965 9418 8569 66694 62378 91474 38547 79773 40661 95805 57953 24051 29911 67172 32462 3431 85790 92129 41033 63565 48129 86431 28365 805 50775 86310 577 41528 22312 80242 8564 39103 8427 26401 26822 55701 81510 2744 34926 70816 134 13786 39401 15735 7792 66087 89023 18058 44409 19305 24256 29373 444 94637 7835 96268 9310 84242 26515 76882 76522 85631 15053 53571 54662 10021 14666 46406 21783 3987 58793 64364 65740 86943 44941 13074 55930 4124 67208 19328 7970 59356 97445 55400 72089 61401 79769 24162 37128 60918 69332 49357 85617 58704 60146 27420 63925 4917 31983 55984 68787 91028 50187 92396 37036 13963 38177 85888 10905

978 6 1

1047 51194 66469 29042 22163 40721 3861 22411 24783 109 52831 94533 3264 60276 98 29575 93701 66152 8295 64237 46736 61405 75052 78759 58517 30750 10150 75808 44377 38956 62672 80093 74820 32455 89393 7003 9389 88033 84210 45298 31738 15964 12921 81377 6111 49132 23907 9530 70864 46722 1796 9381 46047 49196 78692 13194 83762 71335 97153 13752 37357 41617 10831 19895 93366 80354 34473 71135 49083 94785 87590 58356 5911 48495 10869 26341 42521 3089 86397 16267 39259 91615 31585 97359 49046 67307 56860 98289 77145 49135 34899 50395 83070 27924 50867 19833 20472 79838 41057 4550 43190 6292 56251 366 71172 42397 74012 97700 69746 50409 89637 38695 46675 328 15754 59722 89539 22561 17009 59502 27480 11384 47163 50650 8974 59895 4271 80928 9665 27787 34070 84326 18860 39424 47469 27888 4202 42836

36418 69502 1950 84415 48909 38477 30626 60714 65194 52653 71497 78238 40736 24742 76328 82811 47627 59881 6582 80605 73449 68433 29995 91650 82520 46956 65102 35077 6961 24724 16742 82311 60475 47268 51148 45522 25619 84211 92593 96354 56497 58527 73157 93523 60967 19344 48599 55785 91619 47950 88421 46608 7932 47960 63974 20001 19497 31924 23449 24662 3801 44476 90718 63606 58830 77441 97865 2796 49365 86867 94045 5245 24440 35021 6868 64958 20982 40223 42993 11765 97777 85852 20650 24576 71883 67273 85902 10953 66782 58572 40157 16256 6670 46318 61018 85956 38802 51110 94020 11542 13227 91294 30200 28268 31919 6805 92504 60750 68424 24547 87972 75814 13854 72687 63105 35304 66747 71712 63633 77657 56446 51602 17318 66092 35666 68737 47251 64782 25838 90418 33760 24841 84215 8449 23646 31782 96928 59361 52658 34716 26874 84962 7742 36399 3294 21132 91239 8096 48032 74794 83187 50391 69403 20186 12555 92103 74545 91366 42306 17129 69765 9402 81219 3621 4792 88028 81648 6478 16739 93671 63007 35205 2609 91506 62234 56971 87199 61252 225 16658 34141 89225 44878 14995 70896 8211 76734 17282 1095 62176 98604 96381 96686 93319 5929 62818 51003 8260 90753 46146 37491 35334 93376 12474 46941 52795 17099 87192 77127 37267 94434 70370 63832 49898 16868 57153 23236 56239 28024 16981 65298 15208 31188 61094 24253 81825 42021 60929 466 47788 33371 15126 49165 94671 71728 54394 6623 99014 54072 24118 1903 36880 29926 93204 28023 4468 89450 43931 70499 35674 10847 53778 41245 48037 42113 59556 69974 89237 18229 55805 97599 75277 87553 98499 4027 21878 72493 32044 44851 13994 94751 51274 71853 44895 68862 21597 51756 19020 68904 46553 77783 64551 48177 29344 99663 31095 11929 21841 83441 76279 22001 13995 71196 39751 3587 38034 25659 3593 5372 82817 9685 95948 66739 41895 42256 94181 14399 36934 9490 43506 9574 83111 58627 94454 14684 23977 21005 46270 11529 71665 40179 22541 39931 11181 84910 15842 43172 96595 46060 47343 94290 97414 14431 58215 31030 80807 38010 63862 70746 73309 14356 78612 26845 23493 56394 92012 10361 56507 19695 44547 96746 90787 74925 58101 16865 61992 32754 85124 72581 61229 10163 69093 66213 5363 36536 31108 86588

67237 92718 26711 3193 13035 8021 6304 15841 16479 93035 116 35376 84736 6538 72192 57436 31426 43903 54678 55966 1977 11526 6578 48304 48992 16226 79846 16788 73863 34801 31575 34890 66479 48287 93594 4893 13490 19869 1822 25186 79415 69784 78420 76222 1741 13217 19157 63550 6092 85167 86668 70434 52713 23927 36914 63634 9112 78275 92323 59830 14343 9607 81762 28507 22805 60317 18478 5671 6715 49291 22568 1973 62352 89141 90129 24497 38843 22950 25394 74323 30936 2101 3673 29135 88783 62158 96408 44263 16954 81789 7688 10274 38865 3655 40660 13667 60839 80912 48582 27140 53495 35722 87386 6703 65973 88581 55657 47156 4265 81781 17331 62991 51182 95650 97062 92509 8420 87949 28179 27294 38132 52230 90780 42815 17300 24880 2646 74970 23400 85098 63102 75349 48721 77626 2379 56056 33309 60544 8632 48526 36627 10730 50003 31771 43217 77159 95470 16537 63823 42531 13162 50825 49349 47786 47253 68258 53575 75250 79578 11287 29513 85615 53543 82828 44115 70555 55661 53765 48651 99704 42326 87404 67726 41870 43234 69414 25535 6171 32937 76607 36505 54510 39022 96065 70116 99278 35010 55545 77815 20100 88555 58320 63971 88512 53750 63067 76392 3057 28314 9755 55948 94187 1681 51679 74894 91309 71367 85447 56425 63026 24098 68984 45567 95672 5526 47257 72429 42897 66133 11121 63121 30507 17007 26243 64057 41303 22111 31643 79681 17213 99596 67624 89057 54136 24725 5377 7579 16586 4534 50248 91817 47040 2391 70106 30679 21722 84206 59580 12614 13790 29416 75298 22032 57193 6828 19507 7942 51508 63632 6637 68884 92724 97695 80653 11333 72393 18031 74991 7954 62830 23311 90637 15591 46336 37624 59004 25384 4995 42569 13559 79012 59259 52089 78514 7864 71088 5427 37201 5854 96613 30832 51892 6147 93082 21017 44250 20966 56609 87970 38425 95213 56224 93525 41313 3111 40952 54042 43937 13004 90984 5973 12212 96193 65407 42913 8460 1844 66431 89404 35877 1102 59745 49379 94483 52470 64292 54378 46665 28391 85911 10388 13618 26227 53358 2198 37992 31477 67055 30627 57532 88739 53009 27257 66327 51984 66378 31209 68241 31950 9860 44797 36674 74131 75300 61623 56821 33881 13840 27794 84039 14945 69070 96263 52479 77505 44230 85117 91424

6925 84937 66607 73322 44991 8871 18795 90275 99875 8062 71179 63311 24636 73940 47505 60016 42034 75249 82447 20079 5562 76142 63976 79664 3598 83691 68051 30084 52428 17142 2301 97787 34945 34314 66233 56188 66492 83337 47235 21261 57639 14308 7617 95417 69 40926 57703 67919 78987 63912 18126 24533 42327 71894 70513 37717 67873 40856 24066 13159 29690 87498 8837 4373 34541 95132 46961 61145 66471 54170 98681 54239 18448 44728 38429 65766 90123 35520 29664 18605 89947 94586 30026 38142 46791 52331 33930 1711 36280 76663 85113 73554 57189 49468 1469 24125 84301 47172 33661 34251 95486 84216 10749 68183 70062

407 2 1

15457 62632 85497 57217 96355 66405 70756 73079 9984 88463 76445 24970 22005 98338 45606 11772 20475 6369 29054 65702 81851 40688 90859 61348 33361 73599 88660 70728 5757 68959 72578 45155 21453 4007 31555 41258 90361 65037 48297 72681 24520 64899 54754 73861 79323 91732 81501 86994 12734 44409 83844 98458 14694 40127 55780 12952 28163 9465 996 88487 16276 88918 4544 9453 32634 21628 47811 40737 22260 93410 45688 34165 71116 14027 53491 49862 58518 62329 90574 15485 94447 32530 75711 55692 63952 8617 15433 83994 96426 72553 78586 84551 82551 41658 90905 2352 60252 89710 85937 20686 37807 73089 76991 94393 77921 79012 35123 86007 47654 93783 43101 53593 47960 88597 21724 93638 14161 81893 83425 58112 78853 95877 30780 37751 52605 89287 13799 37852 75232 5682 16353 24771 41775 8135 43017 37226 94964 90627 62932 31381 71538 29308 72853 65611 51059 77903 45712 55259 51689 53595 34055 51661 81666 18075 38875 71710 45873 69885 15654 14506 65515 89576 42502 38360 96013 72618 65898 93420 49443 89473 56876 55190 17755 54051 57254 84320 79800 57642 63300 13813 55621 94939 32215 45380 49821 41684 82015 26174 58502 5960 6871 51316 82376 99947 21838 54582 44711 61555 69328 91532 66030 33364 1645 86330 49178 34009 41426 40671 77429 42854 69957 73281 96294 68588 98035 65137 10496 27220 80961 51551 15668 12077 40585 95083 79390 82519 40002 2370 60796 22030 61700 32052 5784 49251 94514 9406 17551 3238 80380 37964 72983 89154

30371 39311 58530 58130 83491 91752 85382 45870 21309 57286 32880 5061 49548 67803 47306 71123 92871 93934 79890 26214 83279 69185 66429 80868 9420 61323 44830 83031 95640 17473 693 36645 19296 48332 89648 35804 27803 53427 2663 44309 52080 69432 31979 81715 2564 34342 82446 85683 88952 56993 57758 17429 13039 24408 59718 64827 61484 74957 98465 24677 55298 62270 1840 12680 30595 94751 77408 58945 53695 76858 44600 61640 27890 84970 84217 86150 17479 46363 33211 59455 37844 8255 59018 37635 82852 10478 8205 70226 44520 62137 68815 12561 41090 42399 59847 58668 16894 50434 91443 48601 88808 28304 56484 64153 66855 3466 72181 45988 9098 29841 48681 4959 58864 92513 44169 11494 61305 9217 66869 8940 93608 76891 46793 69090 88919 19536 65174 97793 91327 34071 90078 41916 48948 58998 93735 66781 71275 82427 57302 41020 41275

Prog.py

```
import numpy
from operator import itemgetter
import sys
import pandas
def sort_values(arr_1, arr_2, kind='quicksort', ascending=False):
    if not ascending:
        return zip(*sorted(zip(arr_1, arr_2), key=itemgetter(0))[::-1])
    return zip(*sorted(zip(arr_1, arr_2), key=itemgetter(0)))
class Library:
    def __init__(self, number, number_of_book, number_of_days_for_registration, books__per_day):
        self.number = number
        self.number_of_book = number_of_book
        self.number_of_days_for_registration = number_of_days_for_registration
        self.books__per_day = books__per_day
    def best_number_of_book(self, first_day_of_scan=0):
```

```
        global total_book, l_number_of_days_for_registration, l_books__per_day, days, books_res
        max_days = days - self.number_of_days_for_registration - first_day_of_scan
        actual_b = list(set(self.number_of_book) - set(books_res))
        current_total_book = numpy.take(total_book, actual_b)
        max_books = max(min(int(max_days * self.books__per_day), len(actual_b)), 0)
        if max_books == 0:
            return []
        arg = numpy.argpartition(current_total_book, -max_books)[-max_books:]
        return numpy.take(actual_b, arg)
    def high_result(self, first_day_of_scan=0):
        global total_book, l_number_of_days_for_registration, l_books__per_day, days
        max_days = days - self.number_of_days_for_registration - first_day_of_scan
        max_books = max(min(int(max_days * self.books__per_day), len(self.number_of_book)), 0)
        current_total_book = numpy.take(total_book, self.number_of_book)
        ind = numpy.argpartition(current_total_book, -max_books)[-max_books:]
        high_result = numpy.take(current_total_book, ind)
        return numpy.sum(high_result)
def sum_total_book(number_of_book):
    global total_book
    return numpy.sum(numpy.take(total_book, list(number_of_book)))
filename = ["books.txt"]
for file in filename:
    with open(file , "r") as f:
        description = f.read().splitlines()
    books, libr, days = list(map(int, description[0].split(' ')))
    total_book = list(map(int, description[1].split(' ')))
```

```
    position = 1
    l_number_of_books = numpy.zeros(libr)
    l_number_of_days_for_registration = numpy.zeros(libr)
    l_books__per_day = numpy.zeros(libr)
    libraries = numpy.empty(libr, dtype=Library)
    for i in range(libr):
        position += 1
        new_l_num, new_num_of_days, new_l_books__per_day = list(map(int, description[position].split(' ')))
        l_number_of_books[i] = new_l_num
        l_number_of_days_for_registration[i] = new_num_of_days
        l_books__per_day[i] = new_l_books__per_day
        position += 1
        number_of_book = numpy.asarray(list(map(int, description[position].split(' '))))
        libraries[i] = Library(i, number_of_book, new_num_of_days, new_l_books__per_day)
    l_book_result = numpy.vectorize(lambda lib: lib.high_result())
    l_total = l_book_result(libraries)
    vect = numpy.vectorize(lambda book_result, number_of_days_for_registration: book_result / number_of_days_for_registration)
    reg_result = vect(l_total, l_number_of_days_for_registration)
    reg_result, lib_sort = sort_values(reg_result, libraries)
    books_res = set()
    result = 0
    with open(file.replace("books.txt", "result.txt"), "w") as f:
        first_day_of_scan = 0
        for i in range(libr):
            actual_lib = lib_sort[i]
```

```
        actual_number_of_book =
actual_lib.best_number_of_book(first_day_of_scan)
        books_res.update(actual_number_of_book)
        first_day_of_scan += actual_lib.number_of_days_for_registration
        if len(actual_number_of_book) >= 0:
            f.write(str(actual_lib.number) + " " + str(len(actual_number_of_book)) +
"\n")
            f.write(str(' '.join(map(str, actual_number_of_book))) + "\n")
        else:
            f.write(str(actual_lib.number) + " 1\n")
            f.write(str(actual_lib.number_of_book[0])+"\n")
    result += sum_total_book(books_res)
    print("Result:", result)
```

Додаток В

result.txt

```
972 398
78379 78398 78400 36742 36875 36087 36991 36115 36040 36121 51693 52183
20902 52420 39208 20482 36378 20422 20802 52551 20443 36199 20287 52268
51877 20096 20671 20821 20303 39155 20728 20423 36553 39272 39960 11707
11511 39427 11531 11882 11481 11644 11004 11643 39744 39317 39773 39819
11417 11308 11210 8181 85271 40226 87323 29988 66840 29994 95510 40205
89352 89348 87350 29946 93434 99569 91376 87359 97506 83141 5312 30022
81083 60594 60592 60581 5452 68765 83280 87186 60563 99474 40283 91276
60768 87180 87397 60545 78971 40297 97646 81265 78969 93300 68724 99444
97392 87147 29801 79251 83352 85405 81310 40041 29799 83042 29796 30123
80976 83020 91567 87113 68678 99396 93236 66594 30135 68632 78865 80911
87484 79291 95241 81342 97287 89547 91142 5114 97745 29691 83414 92167
60402 87024 87015 84951 89038 79328 66507 80829 97196 89573 93096 95142
86949 40431 86946 60321 97179 95128 84885 97784 91025 5632 95745 93063
```

81419 79371 95757 86908 29563 97809 69139 88954 84838 60954 29542 60255 78683 89639 93735 99156 81449 99142 60229 30259 90944 91703 97074 87612 95009 86809 69186 60183 66318 82697 83537 29446 78600 94982 97029 82683 83552 80635 93793 79456 30308 90872 97004 87655 66284 80611 84702 89709 88786 84688 80588 90828 84669 79481 4771 8861 84636 87678 88728 97920 98959 84617 5763 93825 80505 88685 16998 30355 67223 92763 96848 16973 61084 94794 91804 69283 16967 90683 29232 61102 8745 16929 83635 96799 98838 12820 94737 88594 80400 85691 61117 66063 8701 8663 94677 65996 80331 69317 86454 16074 69321 89802 61134 94641 86436 16800 82336 87763 86424 81621 98010 93915 94618 16096 59799 85735 91880 94611 5864 5868 88464 82319 8567 89839 93939 95988 98671 88430 88417 80230 4452 61177 65889 80218 4440 8532 12627 85765 96590 16712 94513 4398 91929 90404 83745 94490 83752 98580 88339 98578 83756 12076 12560 91956 16184 88327 16187 12093 65795 30974 30973 59634 59633 67407 88298 85841 86250 87891 16211 86246 88287 91993 12511 16604 16599 65748 92371 69472 84165 94054 83816 89963 65729 96447 98161 12475 8375 92046 84149 96146 85908 65712 84142 79767 80044 69531 82091 82086 98469 94115 86177 83881 94369 90267 79790 90031 30640 98456 82071 96402 12432 98230 61576 98242 4035 65672 4227 84097 8320 84088 61385 4034 88015 8296 84069 83929 79972 96339 8272 92127 81987 30782 79925 86066 12333 61464 90127 12274 79859 83984 30732 91126 522 396

36133 36151 36276 20065 36285 36517 20032 20107 11714 20373 8022 11277 39306 68638 20709 8895 20215 8559 11867 39229 20364 29705 8186 68826 29260 51801 20710 29909 51964 39821 11875 20248 20467 20991 11339 52304 11697 29876 68909 29685 8152 20154 39102 29670 51720 8738 51887 29666 29145 8472 29403 20134 52558 29125 20397 39577 8200 39384 29842 20954 52360 11813 39109 39891 8430 8016 68727 52336 39738 29009 11647 68937 30888 4044 4255 66943 60354 30084 30067 66146 65797 61058 30090 60811 4931 30872 66518 4051 61369 67543 30366 12566 30769 60741 4241 66981 69033 69235 69234 65940 12759 60095 69042 59938 12758 65675 67262 60855 65770 66565 61467

60190 66570 60660 67560 30555 60260 69062 61274 67320 60876 12285 69191 69369 4323 59935 59854 66780 12641 69370 30839 4080 65862 59962 59735 30485 30774 69112 30830 66743 4092 30806 60585 69405 66071 66103 4168 69164 30550 65870 4750 59925 67355 30506 61544 30539 4973 12832 79392 87587 81440 82805 79749 84161 16576 79373 16250 40457 87925 81002 87559 78701 5629 40443 89576 88280 84882 86462 88985 82844 5582 80057 80744 12181 87960 78680 85590 85591 79449 89532 88296 80801 81332 81510 12188 84440 16821 88687 85924 80720 85627 5038 5528 79791 79244 80706 89014 88001 82878 79817 83320 82081 88494 79210 16023 5477 88888 78648 89433 5789 83288 86294 86441 87004 89055 16531 83627 5092 16947 88207 79153 81191 40231 84372 5402 78631 84998 79559 86822 81144 81611 84469 85710 86519 83668 86799 86148 85239 16886 82498 12124 89327 87785 82966 5355 82046 84545 84734 85017 89317 89318 5342 85038 88771 81902 82045 86337 16389 40146 80948 82643 5183 85783 87230 85182 83017 16155 85072 80205 5277 85122 78974 81703 5929 80435 83033 5243 81018 79692 85836 87132 87875 81004 85829 84699 89648 91967 90107 91965 89898 90840 96033 91934 90621 96006 93943 91886 96762 93931 98805 91997 97038 99120 90929 91824 92850 92638 93001 93812 98379 91720 91717 97133 95794 95781 94869 89976 91691 92033 99879 92817 96714 93041 91667 95100 96706 96313 95732 91606 96797 99794 96145 95692 95687 99782 99781 90744 90552 91568 91564 98358 95652 96876 92779 93600 95642 95635 98232 97682 97681 97200 96494 93115 95615 92197 93560 93137 94485 93531 99641 99315 94492 96550 94238 95502 99597 96850 97548 95497 91380 98276 91158 94185 90088 91370 90432 97332 94556 95304 93354 93266 90682 93318 92495 93283 93287

72 394

15162 15211 15303 15472 43659 44493 43911 44488 44367 44409 43733 43961 44313 44049 44191 41485 40143 79060 79066 36059 81122 8455 4360 87244 29887 52412 60658 59651 88321 89277 98560 20727 89335 80137 87227 60659 86265 96533 83116 84213 99588 29957 68774 29959 92400 5286 97550 68773 36000 90389 97560 99473 85137 8425 66844 93328 81184 95527 66856 97415

95529 96488 81199 81025 93313 68925 91454 78973 87158 52548 94491 95559 89417 30025 78959 4319 81229 87148 80994 89424 82141 84186 88348 96539 91225 99414 11348 5200 81248 52574 93538 90417 39124 91495 96468 59698 85355 87403 98505 66629 30066 36216 65852 89465 94525 99384 5505 81284 36229 95287 68655 85032 83348 66965 78010 97318 91543 66585 87459 66584 89509 5134 93609 97706 60845 11265 11252 93163 66998 98489 89065 5082 30142 82268 96606 89046 87494 30152 29640 29639 79307 39879 94565 11204 88249 16743 99250 84903 83413 99800 12464 8561 93664 98673 91620 11749 79334 80241 5610 96427 69101 52122 60916 91031 84869 16548 16767 91645 39075 93061 16547 92321 97803 95105 52091 5648 65922 93720 94617 39793 84849 99174 60963 96408 88477 99170 36390 88210 20623 20895 65952 81461 95066 77963 90966 93757 29520 84387 94629 97858 81476 4233 84811 84808 11848 82759 87625 30284 84804 65968 81489 85584 95021 92292 81494 4912 99931 92974 81500 4909 4899 67169 61028 78626 98744 39706 39040 81516 90234 41081 97047 61051 87674 90232 96374 91777 90228 92274 90902 5771 80655 78607 86128 82031 30357 86125 66307 94307 40609 97027 12386 88833 83625 39677 97973 95928 79546 86109 89790 83647 94968 67268 4544 8951 36944 84724 40656 83665 86469 94290 94950 29406 83680 79589 93926 85744 4554 94664 84695 20561 66263 65994 12751 16454 12024 98039 78541 5878 99011 96324 80577 5887 94908 96322 5892 89858 29370 61189 79626 20234 81675 81677 67342 16144 83730 36635 85802 88756 84648 96934 90174 51676 51872 98106 89915 92639 60056 78487 91970 8853 94024 84023 88723 51678 91996 80520 29320 84021 61494 4742 89948 86657 98798 40806 96890 77689 4711 8060 78435 81791 96128 98178 8070 69511 69510 51809 90719 77867 4137 59996 88103 39515 39502 4131 8111 8770 16305 85938 67516 36797 92735 81858 67528 90688 96831 59966 51773 81955 40915 82490 61398 88628 29188 78385 83930 36827 59945 83990 77837 80420 94182 61414 94184 98285 8736 16901 90095 80390 4106 88069 78355 96777 8719 99601

602 392

44255 44376 41369 20639 20278 20356 20762 78023 77846 36168 40696 36854 20347 78361 36757 36266 36524 41530 41121 40893 78318 41508 40914 36501 36316 20301 20414 78349 78406 77609 20250 52131 52434 51755 39059 51856 51658 39367 52402 52153 52458 39099 39292 51767 39319 52073 51871 39387 11068 29933 39751 11490 8515 8504 8059 8237 39574 11091 39768 8647 8253 39709 8836 39394 29541 39492 39402 8292 29997 11157 11902 29811 11370 11174 29606 11011 39525 29777 39857 29731 8798 29746 11215 8362 29058 79191 68946 90492 5466 81219 92548 95581 96635 82293 65899 40296 68968 89448 90471 88461 85363 30071 91458 89464 96658 90465 60800 81281 60802 80221 83262 84370 81286 83243 95532 40331 83339 65946 95519 92501 40221 30097 90444 40342 95640 83220 96588 29092 83216 81165 90433 65956 98726 5548 90429 59815 86331 87475 59706 79070 40153 93630 87255 94517 85206 66772 88371 79057 83151 85462 60621 84258 88353 69090 5609 65983 88512 85484 81079 95415 66005 92634 83115 30201 97788 81071 94488 67075 40108 80149 85524 97812 12773 16870 94695 93728 81443 12549 40487 5284 79004 91696 97434 78998 89655 96513 78999 87189 81480 92414 30286 67152 82865 85586 30961 87183 95836 5727 90610 87651 67172 68748 83086 88565 88572 97897 99464 80102 83074 92390 30318 80096 83568 94426 80380 91765 69237 61048 96765 87162 90624 4309 4612 93832 85096 88585 65747 59919 40024 88273 93265 84168 89170 30370 60495 83622 93260 87114 91207 82464 87103 92349 84158 87100 84532 87090 96436 99371 93225 85030 88233 97315 92732 86589 80926 95251 16955 87050 69330 96419 30426 84538 60420 95975 95978 87043 89074 98889 84557 65683 69364 59988 84950 84946 80846 86161 99279 86615 90712 12431 16984 86620 12043 95167 82056 92293 96016 80005 80004 91928 89013 80818 16514 90724 89008 93101 94331 89900 16175 66455 79665 61235 60309 98103 87868 80780 96378 86135 86922 29293 30544 84863 61266 4199 65637 91006 88161 60014 4179 82806 60278 81764 98935 90748 84611 79934 86078 94066 88926 82780 30588 61489 81788 88730 99150 61313 60237 80716 99145 12166 30768 96305 96138 80698 12171 97078 98352 88739 90006 98205 30623 83874 79918 60071 84010 98340 12206 97056 79903 12316 87996

61462 88059 88753 30661 94910 94154 79820 94232 88014 88856 94920 61389 84684 79826 94228 96204 90074 79835 92942 94172 12296 4872 60116 79850 4862 98313 12293 16390 4098 82686 98299

157 390

44532 44571 41349 41326 41500 40620 41436 40684 61375 12614 80198 11402 68740 36670 83772 84304 4024 78975 11381 30670 67393 60590 12602 29879 81009 80183 16731 81000 59745 52415 4454 79672 4393 59755 60624 5206 82213 36855 81111 29059 4389 4381 11492 5348 66630 82206 29763 29757 61358 36089 78907 60667 52478 82315 52277 39988 68874 83211 82317 16780 81876 5169 82986 12104 11292 52250 68628 83234 16645 5923 68627 20255 5414 82321 77783 29711 11276 4500 36153 83261 12538 52234 29077 40268 20889 30626 81234 36178 20898 69591 78248 30048 30683 65753 79209 84399 66923 12506 11630 39922 51633 11241 52200 60787 78822 8008 82137 12047 52189 59824 82135 30931 4103 20912 36235 51638 83341 78796 80072 36771 16826 82369 30110 78277 60834 78278 36261 5028 82382 8656 8659 12194 80362 12119 4984 79288 40706 81336 20385 66420 65720 20660 82416 80050 79306 11099 16560 12785 67485 66043 30886 16218 67030 36311 16904 84128 30619 60246 79325 81819 8341 79327 67042 61590 36317 80718 80722 39061 30864 8332 83439 79766 82460 29509 66078 77801 51743 39456 66083 79381 66365 30237 84795 39035 80436 5876 67110 16950 83498 4906 60974 30256 30602 41076 36404 69362 67434 39021 30266 77803 4896 81478 67142 36950 66125 29267 20043 4703 40519 20048 8973 66315 84745 77905 29280 61516 65608 20070 12900 16453 29287 39682 84586 12356 66296 51831 40583 83594 4851 84722 29311 8944 36588 51950 79742 67229 12347 20133 79929 60130 8066 77879 29316 4828 16056 30716 80526 60053 12951 81936 83845 16078 84686 66212 51879 69334 16260 83677 8882 4799 84955 89829 91872 87724 85657 93803 91701 97833 93732 91893 95780 91679 89608 91631 91609 85453 95679 91577 91913 89525 96013 87442 91502 93543 87373 91442 87344 87337 97574 85277 99604 87312 98100 94004 99578 89283 98105 87235 99494 99479 91268 87169 85106 85051 85825 91973 89146 87878 91176 93193 87042 89069 93152 87005 99211 97156

91991 90969 93006 90957 96093 92998 85858 96099 88897 86834 95019 88872
99104 99103 92018 94999 94996 90894 90864 99053 99052 88803 98179 96940
96925 86678 90760 86633 94780 94779 89997 90675 86576 88617 98851 86554
92694 96779 88569 94110 92658 88555 94692 90554 94648 96669 98216 98694
92512 90454 88399 86342 94155 96573 94503 94484 94482 98564 86252 85977
94431 88261 98495 92341 88220 92285 90214 92262 90208 92252 92222 90169
86051 94211
134 0
882 0
416 0
221 0
541 0
278 0
92 0
629 0
947 0
328 0
761 0
703 0
948 0
450 0
435 0
281 0
275 0
231 0
406 0
628 0
120 0
463 0
762 0

471 0

552 0

856 0

386 0

44 0

70 0

682 0

144 0

13 0

198 0

841 0