ны (во время работы алгоритма вершина может "подниматься вверх"). Высота вершины определяет, куда мы хотим направить избыток.

Алгоритм "проталкивание предпотока" использует две основные операции: проталкивание потока из u в v, и подъем вершины.

Пусть $G = (V, E)$ сеть с истоком s и стоком t, f — предпоток в G, где V — множество вершин, E — множество ребер. Функция h называется *высотной функцией* для предпотока f, если

$$h(S) = |V|, h(t) = 0 \text{ и } h(u) \leq h(v) + 1.$$

1. Проталкивание из вершины u в v возможно, если

1) вершина u переполнена (т.е. $e(u) > 0$);

2) ребро (u, v) не насыщено (т.е. $c_f(u, v) > 0$);

3) $h(U) = h(v) + 1$.

Увеличение потока из u в v ограничено избытком вещества в u пропускной способностью ребра (u, v).

2. Для подъема вершины u на максимальную высоту, допустимую по определению высотной функции, необходимо:

1) вершина u переполнена;

2) для любого ребра $(u, v) \in E_f$ выполнено неравенство $h[u] \leq h[v]$.

Если есть соседняя вершина v, высота которой на единицу ниже, то можно полнить проталкивание (но нельзя выполнить подъем) и, наоборот.

В результате анализа данного алгоритма можно сделать вывод о том, что его достоинствами являются сокращение времени решения задачи за счет нахождения максимума на основе локальных оптимумов и начальный поток берется равным пропускной способности первого разреза.

КЛАССИЧЕСКИЙ АЛГОРИТМ ФОРДА – ФАЛКЕРСОНА

Головко В.В., Маслова З.И.

Известно несколько алгоритмов реализации метода Форда — Фалкерсона отыскания максимального потока, отличающихся временем их работы. Наиболее быстрым алгоритмом является алгоритм "проталкивания предпотока", но он одновременно является и наиболее сложным

в реализации. Более прост в этом отношении классический алгоритм Форда – Фалкерсона. Скорость его работы уступает скорости работы предыдущего алгоритма, но для графов с количеством вершин до нескольких сотен это обстоятельство можно не учитывать, так что простота его реализации является весьма существенным критерием выбора этого алгоритма для решения поставленной задачи. По классическому алгоритму Форда – Фалкерсона на каждом шаге его работы ищется произвольный дополняющий путь p и поток f увеличивается на величину $c_f(p)$ остаточной пропускной способности по пути p. Шаги выполняются до тех пор, пока существуют дополняющие пути.

Этот алгоритм и был положен в основу программы (реализация на Visual C++ 6.0), которая находит максимальный поток для произвольно заданной сети. Вся работа с графом (сетью) реализована в классе *EGraph*, функции-члены которого сгруппированы по выполняемым действиям: задание графа; нахождение максимального потока; вывод графа на экран; служебные функции. Представление графа хранится в виде двух массивов – массива вершин и массива рёбер. Вершины описывает структура *_VERTEX*, а ребра – структура *_EDGE*.

Алгоритм Форда – Фалкерсона реализован функцией *FordFalkerson*, которая в свою очередь вызывает функцию *FindAugmentingPath*, отыскивающую дополняющие пути. Пути находятся с помощью так называемого поиска с возвратом. Для запоминания пройденных вершин и рёбер используются стеки. Выбор рёбер, соответствующей текущей вершине, а также их отсеивание производится совместной работой функции-члена *EnumEdges* и функции обратного вызова *SelectEdgeProc*. Последняя, кроме того, ведёт подсчёт остаточной пропускной способности пути; определяет текущее направление рассматриваемого ребра; производит сокращение встречных потоков, а также занимается установкой специальных статусов вершин и рёбер.