UDC 519.6
DOI: 10.31891/CSIT-2021-4-10

OLHA KUZMENKO, VITALIIA KOIBICHUK,
VALERII YATSENKO, KONSTANTIN HRYTSENKO, ROMAN KOCHEREZHCHENKO
Sumy State University

ALGORITHMS FOR SEARCHING THE SHORTEST PASSES AND THEIR APPLICATION IN THE COMPUTER GAMES

The article examines the application of technologies of artificial intelligence elements in computer games, e-sports. The use of graphs to determine the shortest path for finding elements of electronic games and the use of search algorithms for the shortest path: Dijkstra, BFS, DFS algorithm in artificial intelligence systems, based on which developed a fundamental part of the logic of artificial intelligence to achieve the best gaming experience and justification. on the market. The practical implementation of the search for the shortest and cheapest paths in graphs based on their bypass with the help of search algorithms in width, depth, and Dijkstra algorithm is carried out. The practical implementation of the proposed algorithms is carried out in the Python programming language, the results of which can be integrated into artificial intelligence systems. Implementation of algorithms is made in a procedural style. This allows you to reduce the amount of program code, which gives a clearer idea of the aspects of the implementation of the algorithm. In addition, the result of practical implementation is advanced algorithms. Working prototypes of algorithms find a way from conditional point A to conditional point B, taking into account the target task to define the shortest way or to find the cheapest way. The implementation of algorithms can be the basis for further development, taking into account the reverse operation of the algorithms, to restore the route from the endpoint to the starting point.

Keywords: computer games, e-sports, graph traversal algorithms, the shortest path, artificial intelligence

ОЛЬГА КУЗЬМЕНКО, ВІТАЛІЯ КОЙБІЧУК,
ВАЛЕРІЙ ЯЦЕНКО, КОСТЯНТИН ГРИЦЕНКО, РОМАН КОЧЕРЕЖЧЕНКО
Sumy State University, Sumy, Ukraine

АЛГОРИТМИ ПОШУКУ НАЙКОРОТШИХ ШЛЯХІВ ТА ЇХ ЗАСТОСУВАННЯ В КОМП'ЮТЕРНИХ ІГРАХ

Індустрія кіберспорту пережила значний підйом за останні роки, безперечно, цьому сприяла пандемія коронавірусу в 2020 році, яка змусила скасувати багато спортивних заходів та традиційні спортивні ліги.

Від піонерів комп'ютерних наук до сьогодення ми можемо бачити незрівнянно великий стрибок у розвитку обчислювальних потужностей. Як наслідок, з'явилися нові сфери бізнесу і послуг з якими раніше люди не стикалися: Big Data, наука про дані, Data mining, хмарні технології, 3D- симуляції, штучний інтелект. Захоплення технологіями штучного інтелекту та їх застосуваннями у всьому світі призводить до зростання кількості штучних технологій та розпалює інтерес існуючих компаній у цій галузі. Кількість поглинань стартапів зі штучного інтелекту неухильно зростає. Галузь ігрової індустрії є високорозвиненою сферою, що використовує системи штучного інтелекту.

В статті досліджені питання застосування технологій елементів штучного інтелекту в комп'ютерних іграх, електронному спорті. Обґрунтовано використання графів для визначення найкоротшого шляху щодо пошуку елементів електронної гри та застосування алгоритмів пошуку найкоротшого шляху: алгоритм Дейкстри, BFS, DFS в системах штучного інтелекту, на основі яких розроблена фундаментальна частина логіки ігрового штучного інтелекту для досягнення найкращого ігрового досвіду та обґрунтування успішного комерційного продукту на ринку. Здійснено практичну реалізацію пошуку найкоротших та найдешевших шляхів в графах на основі їх обходу за допомогою алгоритмів пошуку в ширину, в глибину та алгоритму Дейкстри. Практичну реалізацію запропонованих алгоритмів здійснено мовою програмування Python, результати якої можуть бути інтегрованими в системи штучного інтелекту. Реалізація алгоритмів зроблена в процедурному стилі. Це дозволяє зменшити обсяг програмного коду, що дає більш наочне уявлення про аспекти реалізації алгоритму. Крім того, результатом практичної реалізації є розширені алгоритми. Працюючі прототипи алгоритмів знаходять шлях з умовної точки А в умовну точку Б, беручи до уваги цільове завдання визначити найкоротший шлях або ж знайти найбільш дешевий шлях. Реалізація алгоритмів може стати основою для подальшої розробки, з урахуванням зворотної роботи алгоритмів, по відновленню маршруту від кінцевої точки до початкової.

Ключові слова: комп'ютерні ігри, електронний спорт, алгоритми обходу графа, найкоротший шлях, штучний інтелект

Introduction. The origins of e-sports began in 1972 with scientists at Stanford University, who developed the Spacewar game. The rapid development of e-sports and computer gaming began in 2016, in particular, the growth of online broadcasting platforms used to access e-tournaments and individual streams in e-sports. Growth in this segment, firstly, constantly determines the level of intensity of competition between companies that develop data platforms, including Azubu, DingIt, Hitbox, Twitch and YouTube Gaming, and secondly requires the use of innovative technologies Data mining, artificial intelligence, third, it contains a large number of risks and threats [0] (Fig. 1). The eSports industry experienced an unprecedented rise in 2020, in part due to the coronavirus pandemic, which led to the abolition of many traditional sporting events and leagues [0]. In a survey conducted in August 2020 among the leaders of the sports industry worldwide, about 86.3 percent identified simulated e-sports as an industry with very high revenue growth potential (Table 1).

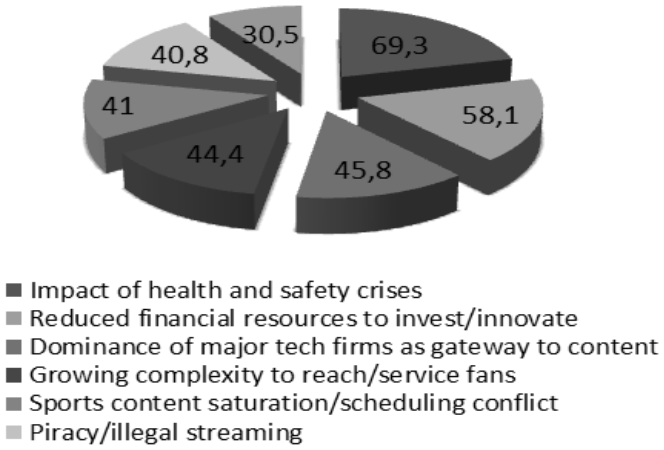

Fig. 1. Threats to revenue in the sports industry worldwide 2020

For example, according to a report by CyberAgent, Inc. [0] in fiscal year 2020, the Japanese Internet media services company generated most sales revenue, approximately 269 billion Japanese yen, through its internet ad business. Next to internet advertising, the company also engages in gaming and investment development: Internet advertisement business – 269,30; Game business – 155.80, media business – 57.

Table 1.

Potential revenue growth in selected sports worldwide 2020

Type of sport	%
eSports (simulated sports)	86.30
eSports (action/fantasy/shooter)	85.80
Soccer	73.60
Basketball	66.40
Urban sports	50.60
Tennis	40.70
Golf	38.20
Cycling	33.30
Motorsport	32
Rugby	31.30
American football	27.70

In addition, the coronavirus pandemic has had a significant impact on the global sports industry: many events have been canceled and professional leagues have been forced to suspend or postpone the season. In a survey conducted in August 2020 [0] among the leaders of the sports industry worldwide, about 69.3 percent of respondents said that the impact of crises on occupational safety and health is a significant threat to the income of the sports industry. Of particular note is the role of artificial intelligence and machine learning algorithms in the development of computer games. The fastest rate of revenue from artificial intelligence in the global market is projected to grow between 2018 and 2027. Although various studies show that global market fluctuations vary. Market research firm IDC predicts that by 2024 the global market for artificial intelligence will reach more than half a trillion US dollars [0]. The current race in the artificial intelligence market is led by IBM, which has a global market share of more than nine percent. IBM is also a leading company in terms of active machine learning and artificial intelligence patents worldwide with more than 5,500 patent families as of November 2020. Alongside IBM in the global patent race for artificial intelligence are Microsoft and Samsung, each within IBM's 500 patent families.

The aim of the work is to develop algorithms for determining the shortest ways to bypass graphs with the possibility of their implementation in artificial intelligence technologies and the development, implementation and maintenance of computer games.

Related works. The urgency of the demand for research in computer game development based on fuzzy logic methods using artificial intelligence tools is confirmed by a significant number of publications. So for the query “computer gaming and machine learning” in the Scopus database, were found 1802 documents published by 4766 scientists over the past five years. Bibliographic analysis of these publications using VOSviewer 1.6.15 Tools allowed to form 15 clusters (Table 2, Figure 2), covering 58 countries with a publishing activity of 547 financial security researchers (minimum number of publications is 3 units).

Table 2.

Division into clusters by country by number of scientific publications 3 or more units on computer gaming and machine learning

Cluster	Country
Cluster 1	Austria, Greece, Ireland, Norway, Russian Federation, Sweden, United Arab Emirates
Cluster2	Egypt, Indonesia, Malaysia, Pakistan, Saudi Arabia,United Kingdom
Cluster 3	Kenya, Netherlands, Romania, South Africa, Thailand,Vietnam
Cluster 4	Brazil, Denmark, Malta, Mexico, Portugal
Cluster 5	Australia, China, Hong Kong, Japan, Qatar
Cluster 6	Colombia, Cyprus, New Zealand, Spain
Cluster 7	Germany, Poland, Switzerland, Ukraine
Cluster 8	Argentina, Israel, Philippines, United States
Cluster 9	Czech Republic, Italy, Slovakia, South Korea
Cluster 10	India, Singapore, Taiwan
Cluster 11	Canada, Chile, Turkey
Cluster 12	Belgium, France, Tunisia
Cluster 13	Bulgaria, Finland
Cluster 14	Iran
Cluster 15	Slovenia

We should also pay attention to the processes of convergence of ordinary games into developing computer games for children using the tools of artificial intelligence in the work of scientists Brana V., Struk O. [0]

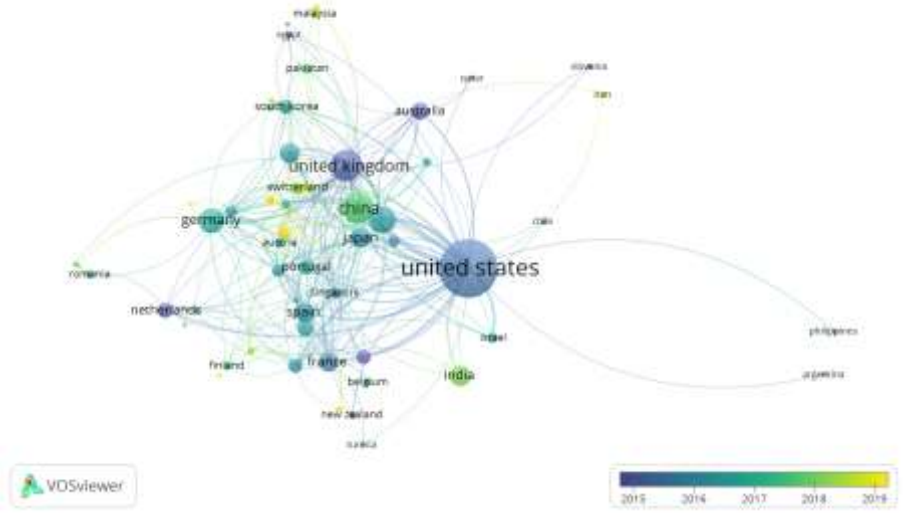

Fig. 2. Bibliometric analysis of scientific publications on the machine learning method and esport industry
Source: developed by the authors on the Scopus data base and VOSviewer 1.6.15 tools

Presentation of the main material. Today, the esports industry is incredibly popular. Profits from the video game industry were estimated at 159.3 billion dollars in 2020, which is 9.3% more than in 2019. Current forecasts estimate the video game industry by 2025 at $ 268 billion.

The largest growth in the eSports industry is in Latin America and the Asia-Pacific region (APAC). Their contribution to economic development increased by 10.3% and 9.9%, respectively, compared to 2019 revenues.

Let's highlight the main factors influencing its development. The first factor characterizes the similarity of e-sports with traditional sports. E-sports is built on competition, as are other sporting events, including football and basketball. Quite often people become fans of a certain type of e-sports: watching competitive gaming events has become a global phenomenon, as e-sports offers interesting stories for big competitions. In addition, like other sports, trade and competition, attracting outsiders are common elements of e-sports. E-sports viewers can enjoy high-level games through online platforms such as YouTube and Twitch. Cyber tournaments often offer easy access and free viewing around the clock. According to the results of a statistical study [0] as of August 2020, the benefits of eSports to rights owners worldwide 2020 are distributed as follows: engaging a new fan base with your sports – 74,3%, attracting new commercial partners – 71,9%, generating new revenue streams – 65,7 %, increasing sponsorship revenues – 61,6%, attracting new participants to your sport – 50,4%, engaging/retaining existing fan base – 49,1%, increasing media revenues – 40,8%. The second factor is High-Quality Games. Video game technology has evolved tremendously since the Magnavox Odyssey, Atari 2600 and Nintendo Entertainment System consoles. Video game graphics, in particular, have become much more realistic, thanks in part to innovations such as 3D and virtual reality (VR), which facilitate the viewing and direct implementation of cyber games. From a

human health point of view, players are less likely to develop eye strain or severe headaches due to pixel graphics. Elements such as augmented reality (AR) can help transform the physical world, so players feel inside the game. The third factor is the high availability and inclusiveness of computer games. People can play video games at any time of the day. This factor is different from many other sports, where the field, gym or skating rink is closed at night. E-sports also do not require much equipment or space to play. If people have a smartphone, they can easily download a mobile game to their phone. Also, a feature of esporty is the ability to succeed without high physical abilities.

So, let's consider the structure of basic algorithms and features of algorithms for finding the shortest way to bypass graphs, which are directly implemented in various methods of artificial intelligence and used in the development of computer games, game design. According to the basic definition, the search for a path is a term in computer science and artificial intelligence, which means a computer program to determine the best, optimal route between two points. That is, the search for a way is widely used in various fields of science and business and video games in this context should be seen not only as entertainment but also as a research platform for studying the effectiveness and operation of such algorithms. First of all, you should determine the data structure that underlies most path search algorithms. Graph - is a set of nodes that have data and are associated with other nodes. The formal definition of a graph is represented by a equation (1):

$$G(V, E) = (V, E), \qquad V \neq \varnothing,\ E \subseteq V \times V, \{v, v\} \notin E,\ v \in V \tag{1}$$

where V – set of vertices (nodes of the graph); E – set of graph edges; v – graph element.

In essence, the path search algorithm searches on the graph, starting from one (starting) point and exploring adjacent nodes until the destination node (end node) is reached. In addition, the path search algorithms in most cases also have a goal to find the shortest path. Some graph search methods, such as width search, can find a way if given enough time. Other methods that explore the graph can reach their destination much faster. There are many such algorithms that can be applied in terms of a specific task or context. Some of the most popular are the following: Dijkstra algorithm, DFS, BFS [0].

Each of these algorithms deserves separate consideration. Each of them finds its application. These are fundamental algorithms, thanks to which there are all the others that perfectly reflect the essence of the search task.

Width Search (BFS) is one method of traversing a graph. Let the given graph $G = (V, E)$ and the selected initial vertex s. The width search algorithm systematically bypasses all edges G to open all vertices achievable with s, calculating the distance (minimum number of edges) from s to each achievable vertex s. The algorithm works for both oriented and non-oriented graphs. Informally describing this algorithm, dividing it into successive steps, you can get the following basic steps: 1) place the initial node from which the search began in an empty queue; 2) remove the node u from the beginning of the queue and mark it as deployed; 3) if this node u was the target, then complete the algorithm; 4) otherwise, at the end of the queue are added all the successors of the node u, not yet deployed, and which are not in the queue; 5) if the queue is empty, then all nodes of the connected graph must have been revised (the target node is inaccessible due to access to the original node); complete the search with the result failure; 6) return to the second step.

The scheme of the graph is shown in fig. 3:

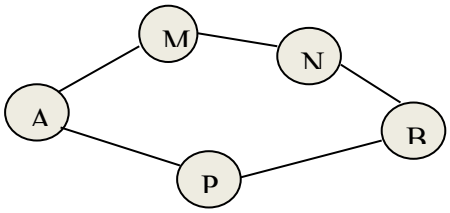

Fig. 3. Organization of data in the form of an undirected graph for the implementation of the search algorithm in width

Using the Python language, this graph (Fig. 3) can be represented as:

```
graph = {'A': ['M', 'P'], 'M': ['A', 'N'], 'N': ['M', 'B'], 'P': ['A',
'B'], 'B': ['P', 'N']}
```

This method of representation is called an adjacency list. Having the following data structure, you can encode the algorithm:

```
def bfs(start_node, goal_node, graph):
queue = [start_node]
    visited = {start_node: None}
    while (queue):
        node = queue.pop(0)
        if node == goal_node:
            break
```

```
            for key in graph[node]:
                if key not in visited:
                    queue.append(key)
                    visited[key] = node
                    print(visited)
        return visited
```

This is a basic bypass algorithm, in order to find the shortest path you need to define another function:

```
def find_shortes(start, goal, visited):
    cur_node = goal
    print(f'path from {start} to {goal}:\n', end='')
    res = [cur_node]
    while cur_node != start:
        cur_node = visited[cur_node]
        res.insert(0, cur_node)

    return " ---> ".join(res)
```

It is important to note that in the first bfs function, the bypass function, we return an object of visited nodes with values indicating the initial node. This will allow you to find the shortest path retrospectively from the target node to the original. In the second function `find_shortes` there are cyclic iterations of return on the shortest way from a target node to initial. To the user-friendly interface, the function uses formatting elements.

As a result of start of algorithm in such configuration we have:

```
start = "A"
goal = "B"
visited = bfs(start, goal, graph)
print(find_shortes(start, goal, visited))
```

Результат:

```
path from A to B:
A ---> P ---> B
```

Let's consider the following depth search algorithm (DFS), which is one of the graph traversal algorithms. The strategy of searching in depth, as the name implies, is to go deep into the graph as much as possible. In general, this algorithm is very similar to the search in width, with only one difference – the choice of nodes. The previous algorithm (BFS) used a data structure – queue (FIFO), which works on the principle of first in, first out (first in, first out), in this algorithm stack – which works on the principle of first in, last out (First-In-Last-Out, FILO), which allows you to not walk on the nearest connected graphs, and follow the inside of the graph, exploring the branch. Otherwise, the algorithms are almost the same. The source code of the algorithm implemented in Phyton is as follows:

```
def dfs(start_node, goal_node, graph):
    stack = [start_node]
    visited = {start_node: None}
    while (stack):
        node = stack.pop()
        if node == goal_node:
            break
        for key in graph[node]:
            if key not in visited:
                stack.append(key)
                visited[key] = node
                print(visited)
    return visited
```

Under the same conditions as in the previous configuration of the width search algorithm, the result will be the same, differing only in the calculation of this path:

```
path from A to B:
A ---> P ---> B
```

However, we have problematic questions: what if the path from one node to another is longer or more expensive? So far, algorithms have been cited as examples that do not consider the cost of relocation, implying that the cost of relocation is the same. These algorithms answer the question "How to find the shortest path?". Algorithms that involve the cost of moving between nodes answer the question "How to find the cheapest way?". The fundamental algorithm is traditionally considered to be the Dijkstra Algorithm. Dijkstra's algorithm is an algorithm on graphs, invented by the Dutch scientist Edsger Dijkstra [0]. He finds the shortest paths from one of the

vertices of the graph to all the others. The algorithm works only for graphs without edges of negative weight.

Based on the formal definition – a weighted oriented graph *G* (*V, E*) without edges of negative weight. It is necessary to find the shortest paths from some vertex a of the graph *G* to all other vertices of this graph. There are a large number of applications in this algorithm, so an example is any problem where there is a process of movement and this process has a cost measurement. For example, it is necessary: 1) to construct a route from point A to point B, using at the same time the shortest route (the price of movement here is time spent on a route); 2) there are a number of flights between cities around the world, for each known cost. The cost of a flight from A to B may differ from the cost of a flight from B to A. In the problem you need to find the minimum cost route.

It should be emphasized once again that this algorithm implements the bypass of the graph, which allows you to find the cheapest route. As in previous search algorithms (BFS, DFS), some modification is required to find a specific path. As in previous implementations, the path will be retrospectively restored by linking nodes that are connected to the original vertex.

You need to start the implementation by importing the required data structure – a heap (specialized data structure such as a tree). This data structure allows generalizations to the priority queue, the priority in the context of this task is determined by the cost of moving:

```
from heapq import *
graph = {'A': [(2, 'M'), (3, 'P')],
         'M': [(2, 'A'), (2, 'N')],
         'N': [(2, 'M'), (2, 'B')],
         'P': [(3, 'A'), (4, 'B')],
         'B': [(4, 'P'), (2, 'N')]}
```

Having the tools of this library and a weighted graph, you can define a function:

```
def dijkstra(start, goal, graph):
    queue = []
    heappush(queue, (0, start))
    cost_visited = {start: 0}
    visited = {start: None}
    while queue:
        cur_cost, cur_node = heappop(queue)
        if cur_node == goal:
            break
        next_nodes = graph[cur_node]
        for next_node in next_nodes:
            neigh_cost, neigh_node = next_node
            new_cost = cost_visited[cur_node] + neigh_cost
                  if neigh_node not in cost_visited or new_cost <
cost_visited[neigh_node]:
                heappush(queue, (new_cost, neigh_node))
                cost_visited[neigh_node] = new_cost
                visited[neigh_node] = cur_node
    return visited
```

Now with the help of the find_shortes function, which has already been declared in the BFD algorithm, we can get the cheapest way:

```
start = "A"
goal = "B"
visited = dijkstra(start, goal, graph)
print(find_shortes(start, goal, visited))
```

Результат:

```
path from A to B:
A ---> M ---> N ---> B
```

The obvious question is: how does gaming artificial intelligence find a player? Computers are not people, they can't understand the simplest directive for people like "Come to me" or "Go to the store", just as people don't understand "01000011 01101111 01101101 01100101 00100000 01110100 01101111 00100000 01101101 01100101", although it is mostly meaningless in the context of a game developer. The task of creating a game becomes real when using high-level languages, writing code, which will later be compiled or interpreted to another level of abstraction and transmitted to the computer in the form of binary code. Instead of kilometers of binary code, it is enough to develop an appropriate function that will teach the computer to determine the path of finding the player or elements of the game. Thus, the task of finding a path consists of two stages: adapting the game world to a mathematical model and finding in this model the path between two points.

Conclusions. According to the results of the study, it can be argued that the basic algorithms for traversing the graph and finding the path involve a simple logic that requires power to calculate. These algorithms are

fundamental, because their combination, modification generates new algorithms that are more specific to a particular logic or task. There are many tasks, for example, from the standpoint of logic to determine the optimal routes, from the standpoint of artificial intelligence of the car's autopilot, objects in the simulation or the ghosts of Pinky from the arc video game Pac-Man. At the heart of these systems is the responsible task of finding a way, sometimes it depends on life, as in the case of autopilot, or gaming experience, which made the usual game a bible for game designers. The study demonstrates the practical implementation of three algorithms for finding the shortest paths, namely the traversal of the graph in width, the traversal of the graph in depth and the Dijkstra algorithm, which is the basis in artificial intelligence systems for the development of computer games.

References

1. Sports Industry: system rebooting. URL: https://www.pwc.ch/en/publications/2020/PwCs-Sports-Survey-2020.pdf (date of access: 13.03.2021). – Title from screen

2. Sport Industry : System Rebooting : Pwc's Sports Survey 2020 / Pricewaterhousecoopers. URL: https://library.olympics.com/Default/doc/SYRACUSE/470563/sport-industry-system-rebooting-pwc-s-sports-survey-2020-pricewaterhousecoopers?_lg=en-GB (date of access: 07.03.2021). – Title from screen

3. Integrated Report : CyberAgent Way 2020 (Integrated Report). URL: https://www.cyberagent.co.jp/en/ir/library/annual/ (date of access: 13.03.2021). – Title from screen

4. Artificial Intelligence (AI) market size/revenue comparisons 2018-2027. URL: statista.com

5. Brana V.Yu., Struk O.O. Vykorystannia shtuchnoho intelektu v rozvyvaiuchykh kompiuternykh ihrakh: Innovatsiini tekhnolohii tsyfrovoi osvity u vyshchii ta serednii shkoli Ukrainy ta krain Yevrosoiuzu. URL: http://dspace.tnpu.edu.ua/bitstream/123456789/14506/1/Brana.pdf (date of access: 13.03.2021). – Title from screen

6. Perceived benefits of eSports for rights owners according to sports industry leaders worldwide as of August 2020. URL: https://www.statista.com/statistics/1192789/esports-rights-owners-beneifts/ (date of access: 13.03.2021). – Title from screen

7. Sholom P.S., Zdolbitska N.V. Analiz alhorytmiv obkhodu hrafa dlia zadachi trasuvannia marshrutu / P.S. Sholom, N.V. Zdolbitska // Mizhvuzivskyi zbirnyk «Kompiuterno-intehrovani tekhnolohii: osvita, nauka, vyrobnytstvo». – 2011. Vyp. # 3. – S. 204-207.

8. Dijkstra's Algorithm in Python. URL: https://stackabuse.com/dijkstras-algorithm-in-python/ (date of access: 13.06.2021). – Title from screen

Olha Kuzmenko **Ольга Кузьменко**	D.Sc., Professor, head of the Economic Cybernetics Department, Sumy State University, Sumy, Ukraine e-mail: o.kuzmenko@biem.sumdu.edu.ua orcid.org/0000-0001-8520-2266, Scopus Author ID: 57214957739, Researcher ID: W-5504-2019 https://scholar.google.com/citations?user=Gpd0708AAAAJ&hl=ru	доктор економічних наук, професор кафедри економічної кібернетики, Сумський державний університет, Суми, Україна
Vitaliia Koibichuk **Віталія Койбічук**	Ph.D, Associate Professor of the Economic Cybernetics Department, Sumy State University, Sumy, Ukraine e-mail: v.koibichuk@uabs.sumdu.edu.ua orcid.org/0000-0002-3540-7922, Scopus Author ID: 57204028825, Researcher ID: AAV-3030-2021 https://scholar.google.ru/citations?hl=ru&pli=1&user=DGI-7VYAAAAJ	кандидат економічних наук, доцент кафедри економічної кібернетики, Сумський державний університет, Суми, Україна
Valerii Yatsenko **Валерій Яценко**	Ph.D, Associate Professor of the Economic Cybernetics Department, Sumy State University, Sumy, Ukraine e-mail: v.yatsenko@uabs.sumdu.edu.ua orcid.org/0000-0003-2316-3817, Scopus Author ID: 57208010199, Researcher ID: P-5487-2014 https://scholar.google.com.ua/citations?user=oOkb-48AAAAJ	кандидат технічних наук, доцент кафедри економічної кібернетики, Сумський державний університет, Суми, Україна
Konstantin Hrytsenko	Ph.D, Associate Professor of the Economic Cybernetics Department, Sumy State University, Sumy, Ukraine e-mail: k.hrytsenko@uabs.sumdu.edu.ua orcid.org/0000-0002-7855-691X, Scopus Author ID: 6601977173 https://scholar.google.com/citations?hl=ru&user=CrvuGuwAAAAJ	кандидат технічних наук, доцент кафедри економічної кібернетики, Сумський державний університет, Суми, Україна
Roman Kocherezhchenko **Костянтин Гриценко**	applicant, Sumy State University, Sumy, Ukraine e-mail: r.kocherezhchenko@student.sumdu orcid.org/ 0000-0001-7269-4177 https://scholar.google.ru/citations?view_op=list_works&hl=ru&user=xO9lHs4AAAAJ	здобувач освіти, Сумський державний університет, Суми, Україна