National Research University Higher School of Economics (Higher School of Economics/HSE)

Faculty of Computer Science

Bachelor's Programme Data Science and Business Analytics

01.03.02 Applied Mathematics and Computer Science

Individual Internship report

Fulfilled by

Pavleeva Maria Valeryevna

Internship goals : consolidation, expansion and deepening of the theoretical knowledge and acquirement of initial practical skills in solving specific problems

Tasks:

a. consolidation and deepening of the theoretical knowledge on the disciplines passed at the university;
b. acquirement of information competence for the purpose of successful work in professional activities;
c. obtaining skills of both independent and team work.

Contents : My part of the project work was the implementation of the Karger's algorithm for finding the cut of the minimum weight. In our project, this algorithm is used to find subgroups in the «Квесчн» group of VK, selected by us.

Educational Internship Schedule (Plan):

№	**Calendar period**	**Work Plan**	Internship Supervisor's signature/
1	01.07.2019	1. Organizational (induction) meeting	
2	01.07.2019	2. Instructing on the requirements of labor protection, safety, fire safety and internal labor regulations	
3	01.07.2019 - 13.07.2019	3. Fulfillment of Individual Assignment	
4	01.07.2019 - 13.07.2019	4. Consultation	
5	14.07.2019	5. Preparation and submission of the Report	

Learned materials :

To implement the Karger's algorithm, we decided to use the Python programming language.

So, I decided to use the clarification of the Karger algorithm and its implementation in C ++ (as a sample), presented on the second site (bibliography) to implement the algorithm in Python. However, I had problems when I approached the moment of combining the last and the previous vertices into one.

I recalled the Python programming language, learned Karger's algorithm for finding the minimum cut, and recall part of the Discrete Mathematics course related to graphs.

Description of the results: After several unsuccessful attempts to correct the resulting code, which was based on this, Kristina Smotrova and I could together implement the Karger algorithm for connected graphs, but our graph is not connected, the algorithm is not suitable for disconnected graphs, so it cannot be used for the initial partition.

Conclusion: Practice was really helpful for me, as I recalled some programming skills on Python and C++ and the course of Discrete Mathematics, related to graph theory.

Bibliography:

I used the following internet resources to write the algorithm:

1) https://www.geeksforgeeks.org/kargers-algorithm-for-minimum-cut-set-1-introduction-and-implementation/ - On this site I found an example of the algorithm and one of its implementations on C ++
2) https://e-maxx.ru/algo/stoer_wagner_mincut - the implementation presented on this site came up more suitable for me, since the adjacency matrix is used here to represent the graph

Supplementary material:

```
maxn = 500
n = int()
g = [[0]*maxn for _ in range(maxn)]
best_cost = 1000000000
best_cut = list()

def mincut():
    v = [[] for _ in range(maxn)]
    for i in range(n):
        v[i].append(i)
    w = []
    exist = [True* maxn for _ in range(maxn)]
    in_a = []
    for ph in range (n - 1):
        for i in range(len(in_a)):
            in_a[i] = False
        for i in range(len(w)):
            w[i] = 0
        for it in range (n - ph):
            sel = -1
            for i in range (n):
                if ((exist[i] == True) and (in_a[i] == False) and (sel == -1 or w[i] > w[sel])):
                    sel = i
            if (it == n - ph - 1):
                if (w[sel] < best_cost):
                    best_cost = w[sel]
                    best_cut = v[sel]
                v[sel - 1].insert(v[sel - 1].end(), v[sel].begin(), v[sel].end())
                for i in range (n):
                    g[sel - 1][i] += g[sel][i]
                    g[i][sel - 1] += g[sel][i]
                exist[sel] = False
            else:
                in_a[sel] = True
                for i in range (n):
                    w[i] += g[sel][i]
```