FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
OF HIGHER EDUCATION
ITMO UNIVERSITY

Report

on the practical task 8

“Practical analysis of advanced algorithms”

Performed by

Ilya Lyalinov

Academic group J4134c

Accepted by

Dr Petr Chunaev

St. Petersburg

2021

Goal

Practical analysis of advanced algorithms

Problem formulation

Analyse Prim's and Edmonds-Karp algorithms in terms of time and space complexity, design technique used, etc. Implement the algorithms and produce several experiments. Analyse the results.

Brief theoretical part

Let $G = (V, E)$ be a graph

Prim's algorithm:
It is an algorithm that finds a minimum spanning tree for a weighted undirected graph:
1) Pick any vertex and initialize a tree with it.
2) Add an existing edge (u, v) such that its weight is minimal for all $v \in \{V \setminus T\}, u \in T$; add v to T
3) Repeat step 2 until all vertices from V are in the T

Time complexity estimation:
$O(|E| * lg(|V|))$ if we use binary heap to store vertices from $\{V \setminus T\}$
Memory complexity estimation:
$O(|V| + |E|)$ if we use adjacency list for graph representation.

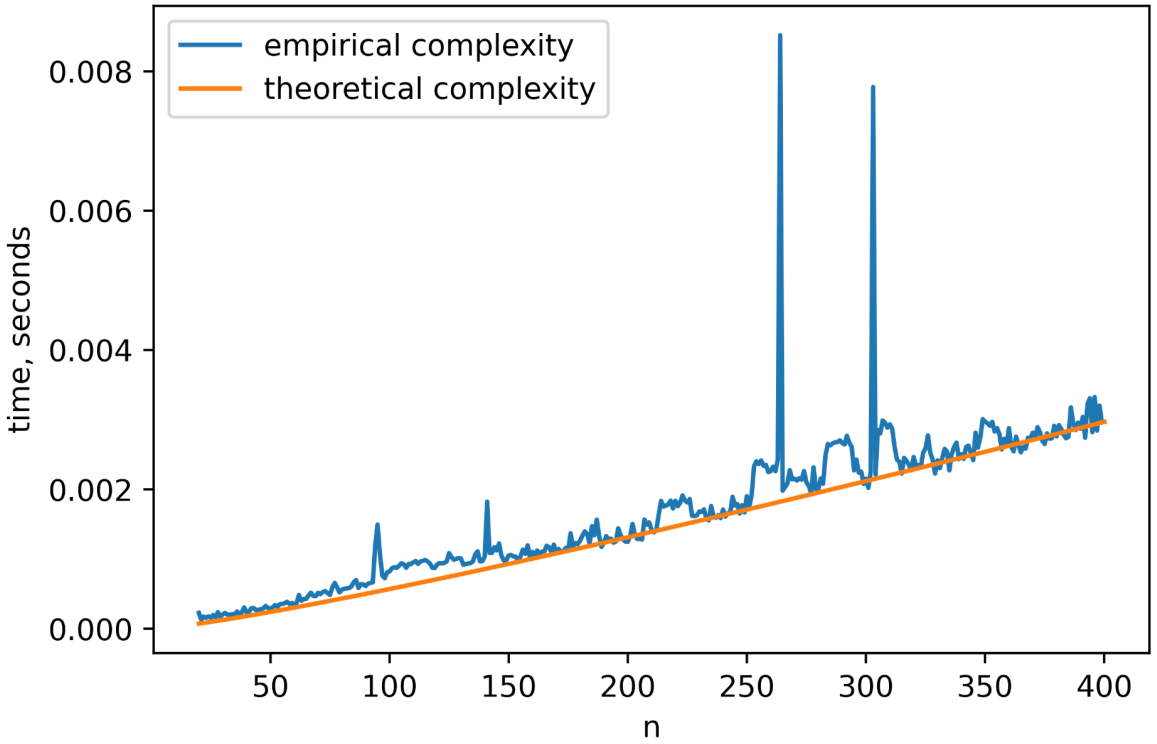
Edmonds–Karp algorithm:
It is an algorithm that finds maximum flow for a weighted directed graph. Each weight of an edge represents its maximum flow:
1) Set the initial value of the flow for all edges to zero.
2) find a path from source to target that increases the current flow. Increase the flow by c_{min}, where c_{min} is the critical edge's capacity in the found path. The increasing path is obtained via BFS.
3) Repeat step 2 until there is no increasing path.
Time complexity estimation:
$O(|V| * |E|^2)$

Space complexity estimation:
$O(|V| + |E|)$ if we use adjacency list for graph representation.

Results

Prim's algorithm:

We created graphs with n vertices and $5n$ random edges where $n = 20, 21, \ldots, 400$. Edges weights were chosen randomly as well. We executed Prim's algorithm for each graph to find minimum spanning tree and measured the execution time. We plotted the dependence of running time of algorithm on the number of vertices (n). We also visualized the theoretical curve ($O(|E| * lg(|V|))$) on the same chart to compare empirical and theoretical time complexity estimates:



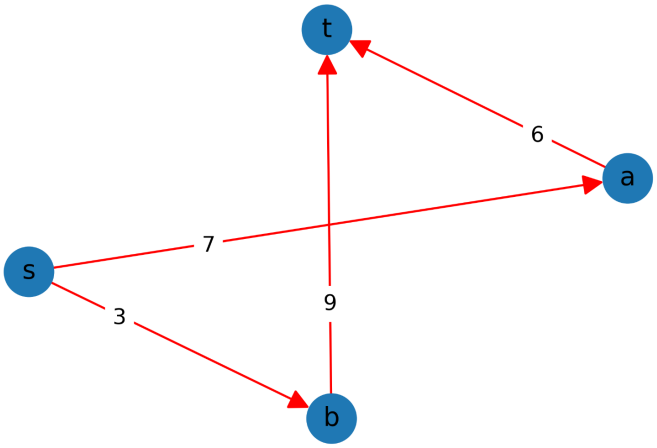
There are no major discrepancies between the curves, apart from some visible spikes. However, these spikes can be explained by the peculiarities of the work of a particular operating system and hardware components of the computer on which the launch of the algorithm was made. This means that the empirical execution time of the algorithm is consistent with theoretical estimates.

Data structures used by the algorithm:

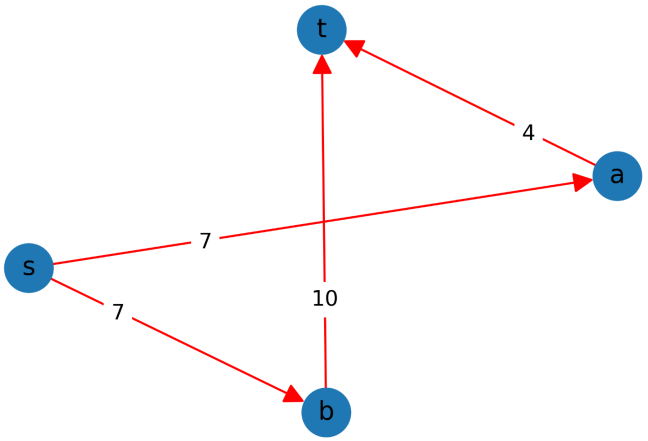
Binary heap for storing vertices $v \in \{V \setminus T\}$; an array or a list for storing minimum spanning tree's edges; adjacency list or matrix for graph representation.

Edmonds–Karp algorithm:

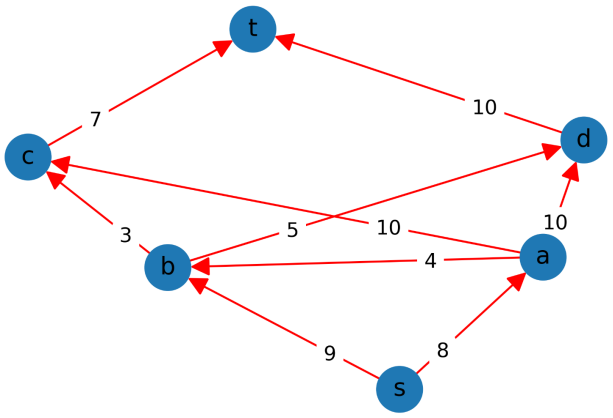
We executed this algorithm on three input graphs. The weights of graphs' edges were chosen randomly from the range [3, 10]. Some graphs have been chosen so that one can easily check that the algorithm works correctly and actually finds the maximal flow. We visualized the graphs and calculated the maximum flow from s to t. (edge labels are the maximum capacity):



$flow_{max} = 9$



$flow_{max} = 11$



$flow_{max} = 16$

We can conclude that the solution obtained by the algorithm is optimal for every graph.

Data structures used by the algorithm:

Adjacency list for graph representation; all data structures for BFS algorithm, since it is used internally in Edmonds-Karp's algorithm (priority queue for vertices); hashtable for storing current flow values at each step for every edge. Adjacency list for residual networks. A list or an array for increasing path at each algorithm's step.

Design techniques used within the algorithms:

Prim's algorithm uses greedy approach. Edmonds-Karp's algorithm is an iterative algorithm.

Conclusion

We have chosen Prim's and Edmonds-Karp's algorithms. We executed it on some generated random graphs and analyzed the results. We observed used data structures and design techniques that were used within the algorithms. We analyzed the algorithms in terms of execution time and space complexity.

Appendix

https://github.com/ilyalinov/Algorithms