FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION ITMO UNIVERSITY

Report
on the practical task No. 8

“Practical analysis of advanced algorithms”

Performed by
Kozlov Alexey
Dmitriy Koryakov
J4133c
Accepted by
Dr Petr Chunaev

St.Petersburg
2022

Table of Contents

Goal

Practical analysis of advanced algorithms

Problems

I. Choose two algorithms (interesting to you and not covered in the course) from the above sections of the book. Graph algorithms, Dinic and Edmonds-Karp, were chosen to solve the maximum flow problem.

II. Analyze the chosen algorithms in terms of time and space complexity, design technique used, etc. Implement the algorithms and produce several experiments. Analyze the results.

Brief theoretical part

In the maximum flow problem, we are given a directed network G = (V, E) with a specified "source" node s and "sink" node t. Each link (i, j) has a capacity u_{ij} (Costs are irrelevant in the maximum flow problem). The objective is to ship the maximum amount of flow from s to t while respecting the link capacities.

Dinic's algorithm

Dinic's algorithm uses following concepts:

- A flow is maximum if there is no s to t path in residual graph.
- BFS is used in a loop.

In Dinic's algorithm, we use BFS to check if more flow is possible and to construct a level graph. In the level graph, we assign levels to all nodes, the level of a node is the shortest distance (in terms of number of edges) of the node from source. Once the level graph is constructed, we send multiple flows using this level graph.

1. Initialize residual graph G as given graph.
2. Do BFS of G to construct a level graph (or assign levels to vertices) and also check if more flow is possible.
   1. If more flow is not possible, then return Send multiple flows in G using level graph until blocking flow is reached. Here using level graph means, in every flow, levels of path nodes should be 0, 1, 2…(in order) from s to t.

A flow is Blocking Flow if no more flow can be sent using a level graph, i.e., no more s-t path exists such that path vertices have current levels 0, 1, 2… in order.

Time Complexity: $O(V^2E)$.
Space complexity is $O(E+V)$.

Edmonds–Karp algorithm

Edmonds–Karp algorithm is an optimized implementation of the Ford–Fulkerson method for computing the maximum flow in a flow network in $O(VE^2)$ time instead of O(E |max_flow|) in the case of the Ford-Fulkerson algorithm.

The algorithm is identical to the Ford–Fulkerson algorithm, except that the search order when finding the augmenting path is defined. The path found must be the shortest path that has available capacity. This can be found by a breadth-first search, where we apply a weight of 1 to each edge.

Time Complexity: $O(E^2V)$.
Space complexity is O(E+V).

Ford-Fulkerson Algorithm

Given a graph which represents a flow network where every edge has a capacity. Also given two vertices source “s” and sink “t” in the graph, find the maximum possible flow from s to t with following constraints: Flow on an edge doesn’t exceed the given capacity of the edge. Incoming flow is equal to outgoing flow for every vertex except s and t.

1. Start with initial flow as 0.
2. While there is an augmenting path from source to sink.
   Add this path-flow to flow.
3. Return flow.

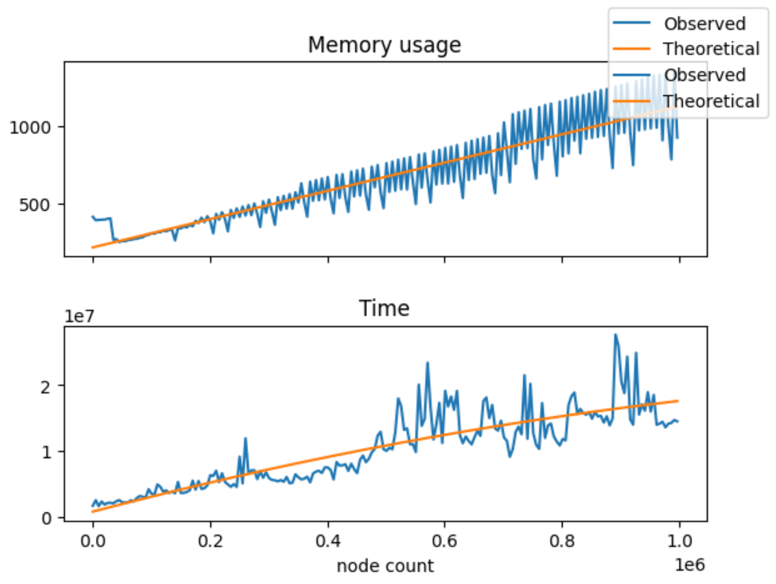
Analysis of time and space complexity.

Dinic’s maximum flow algorithm.

<u>Graphs with complexity of time and memory depending on the number of nodes.</u> The number of edges is constant, E = 10000.

Theoretical time Complexity: $O(V^2)$ if the number of edges is constant.
Space Complexity: O(V) if the number of edges is constant.

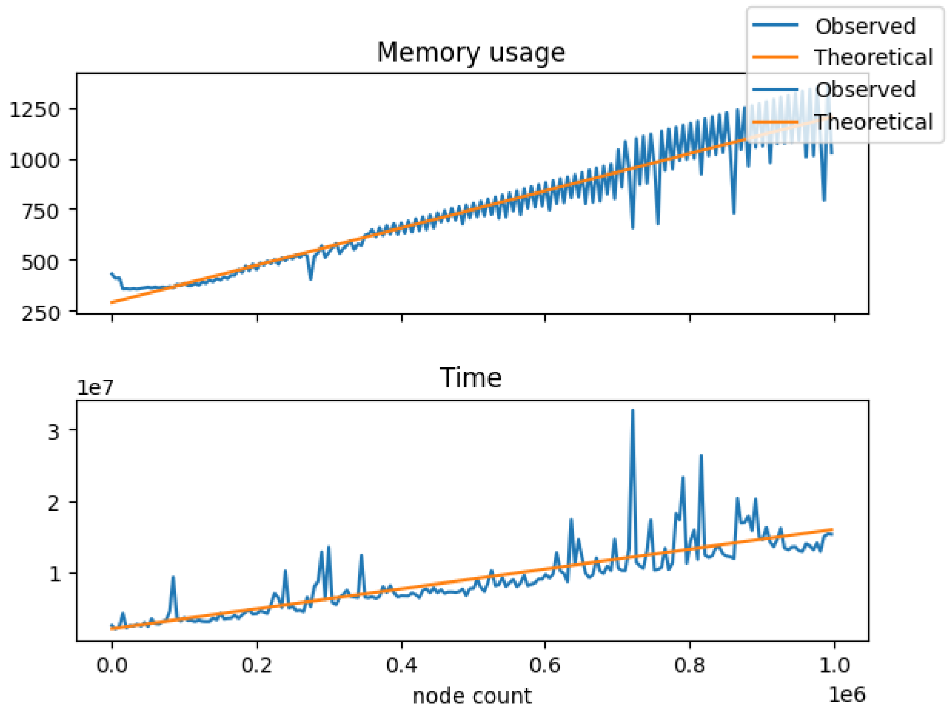
Empirical and Theoretical values are the same.



Graphs with complexity of time and memory depending on the number of edges. The number of nodes is constant, 1000.

Theoretical time Complexity: O(E) if the number of nodes is constant.
Space Complexity: O(E) if the number of nodes is constant.

Empirical and Theoretical values are the same.

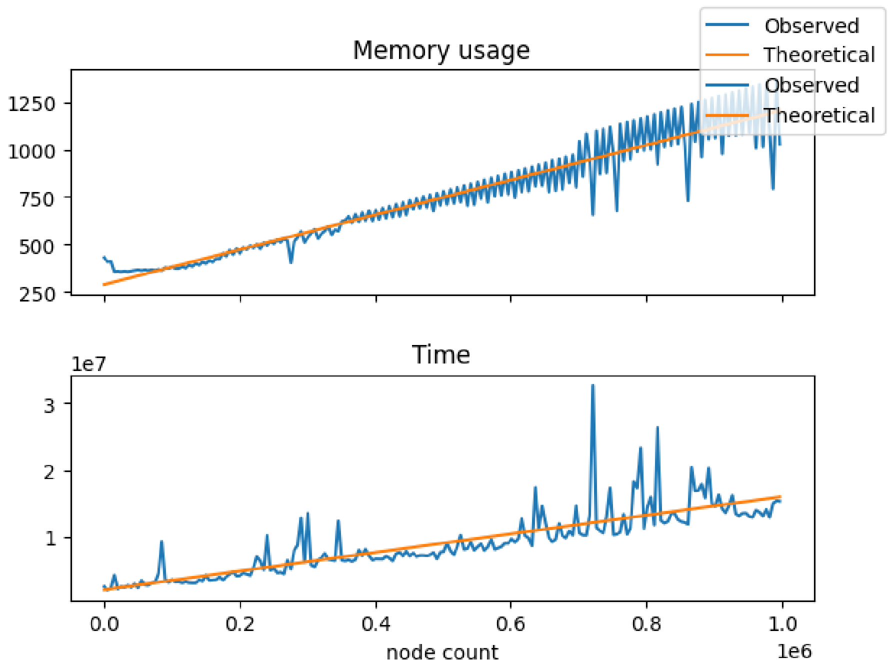


Edmonds Karp maximum flow algorithm

Graphs with complexity of time and memory depending on the number of nodes. The number of edges is constant, 10000.

Theoretical time Complexity: O(V) if the number of edges is constant.
Space Complexity: O(V) if the number of edges is constant.

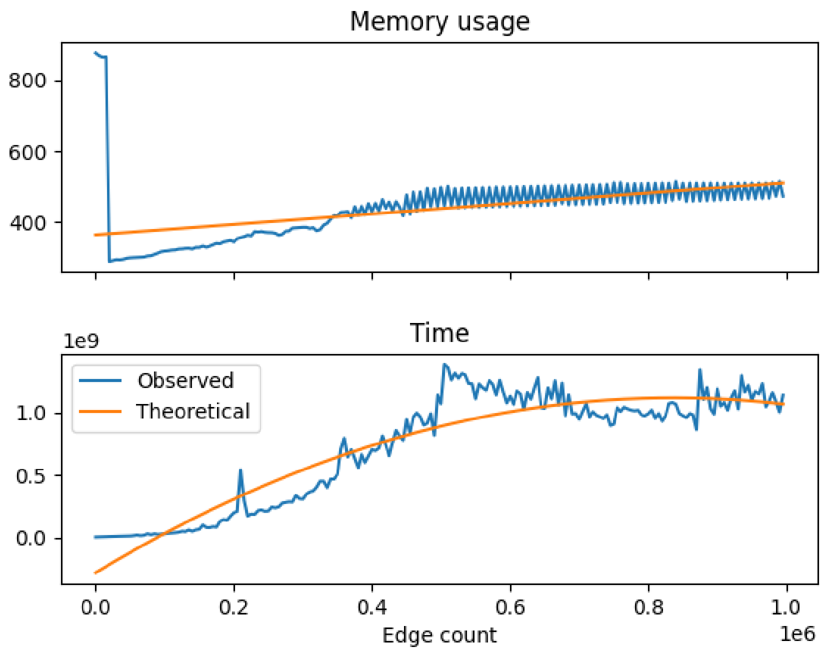
Empirical and Theoretical values are the same.



Graphs with complexity of time and memory depending on the number of edges. The number of nodes is constant, 1000.

Theoretical time Complexity: O(E^2) if the number of nodes is constant.
Space Complexity: O(E) if the number of nodes is constant.

Theoretical and empirical complexity are the same.



Conclusions

Dinic's algorithm

- Doing a BFS to construct a level graph takes O(E) time.
- Sending multiple more flows until a blocking flow is reached takes O(VE) time.
- The outer loop runs at-most O(V) time.
- Therefore overall time complexity is O(V^2E).

Space complexity is O(E+V).

Theoretical and empirical time, spacy complexity are the same.

Ford-Fulkerson algorithm

The running time of O(VE2) is found by showing that each augmenting path can be found in O(E) time, that every time at least one of the E edges becomes saturated (an edge which has the maximum possible flow), that the distance from the saturated edge to the source along the augmenting path must be longer than last time it was saturated, and that the length is at most V.

Space complexity is O(E+V).

Theoretical and empirical time, spacy complexity are the same.

https://github.com/OnlyOneUseAcc/algorithms-RD-practise