ISTANBUL TECHNICAL UNIVERSITY FACULTY OF COMPUTER AND INFORMATICS ENGINEERING

BLG336E – Analysis of Algorithms II
Spring 2022

Lecturer: Assoc. Prof. Dr. İlkay Öksüz
T.As: Res. Asst. Uğur Ayvaz,
Alperen Kantarcı,
Caner Özer,
Muhammed Raşit Erol

Project 3

Mustafa İzzet Muştu
504211564

Development and Environment

The project is coded on Microsoft Visual Studio Code in a Ubuntu 22.04 system. It is compiled by using g++ both on local computer and ITU SSH server without any warning or error. C++11 standard is used and makefile is created. Program can be compiled with "make all" command.

Pseudocode and complexities

Let's say V represents vertices, E for edges, n is number of vertices and m is number of edges in the graph. 'p' is the number of non-zero removal probabilities. Finding combinations of an array has O(p^r) time complexity if r < r – n and O(p^(n-r)) if n – r < n. We can say upper bound for this is 2^p. There is 2^p number of combinations. Let's call it P. Ford-Fulkerson algorithm takes nm^2 time using BFS. However, since there are edge removals, this is an upper bound. If we say all of the combinations have m edges, total time complexity is roughly equals O(2^p + P*(p + p*p + nm^2) + 2^p).

```
FordFulkersonProbabilities (Graph(V,E) G, Probabilities(E), start, end):
    Find non-zero removal probabilities from probability matrix and create an array (P)
    which includes edges with removal probabilities
    C = All combinations of P
    Create temporary graph TMP
    Create temporary combination array CTMP
    Create map FP with flows as keys and probabilities are values
    For each combination c in C:
        TMP = G
        Prob = 1.0
        For each edge and probability in c:
            Remove edge from TMP
            Push c to CTMP
        For each element p in P:
            Exist = false
            For each element in x CTMP:
                If p == x:
                    Exist = true
                    Break
            If Exist == true:
                Prob *= removal probability value in p
            Else:
                Prob* = 1.0 - removal probability value in p
        Clear CTMP
        Flow = FordFulkerson(TMP, start, end)
        FP[Flow] += Prob
    Expected = 0
    For each element m in FP:
        Expected += m.key*m.value
```

Questions

1) How is network flow related to the graph centrality measures?

We can think of a network as a description of the path through which something flows. This allows characterization based on the nature of the flow and the nature of the path encoded by the centrality. Flows can be based on transfers where each indivisible item moves from one vertex to another vertex, such as a good delivery from a city to another city.

2) Describe a potential real-life scenario where this sort of stochastic graph modelling and expected maximum flow can be useful.

We can describe a potential real-life scenario as transportation on highway roads. For example, there are 2 cities where weather conditions are tough between them and there are some roads which can be closed due to too much snow. Or it may be water pipes instead of highway roads. Let's say 's' is the starting point where water distribution starts and 't' is target city. If weather is too cold (under certain degree), some pipes may be frozen.

3) Does adding or removing edges always alter the maximum flow? If yes, please describe the reason. If not, state your objection with a counterexample.

No, it does not always alter the maximum flow. Maximum flow can change only if minimum s-t cut changes. To change minimum s-t cut, we need to increase the capacities on the minimum s-t cut or add edges that can change minimum s-t cut. For example, if we add an edge from vertex 3 to vertex 2 in the given input in the homework, maximum flow does not change.