Algorithms – Homework 4

Due Date: 10.12.22

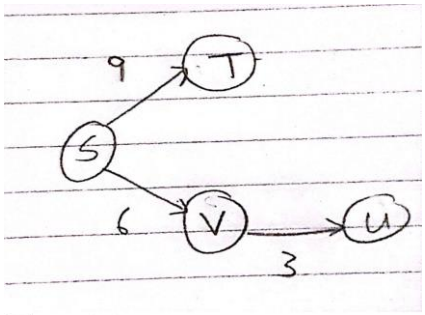
NOTES: 1. the general homework guidelines published in homework 1 are valid for all homework assignments in this course.
2. We will cover the material required to solve problem 5 next week.
3. Note the unusual submission day – Saturday, 10.12.22. No late submission will be approved. The solution will be published on Sunday morning so you have it before the mid-term.

Problem 1 (16 pts.)

In each of the following questions, determine if the stated claim is true (and prove it) or false (and provide a counterexample).

- Let $G = (V, E)$ be a directed graph and let w be a non-negative weight function on the edges. Assume all the weights are distinct, that is $w(e) \neq w(e')$ for all $e \neq e' \in E$.
  Claim: When alg. Dijkstra is performed on G and vertex u is extracted from the queue, then u is the only vertex with minimal d(u).

False, counterexample:



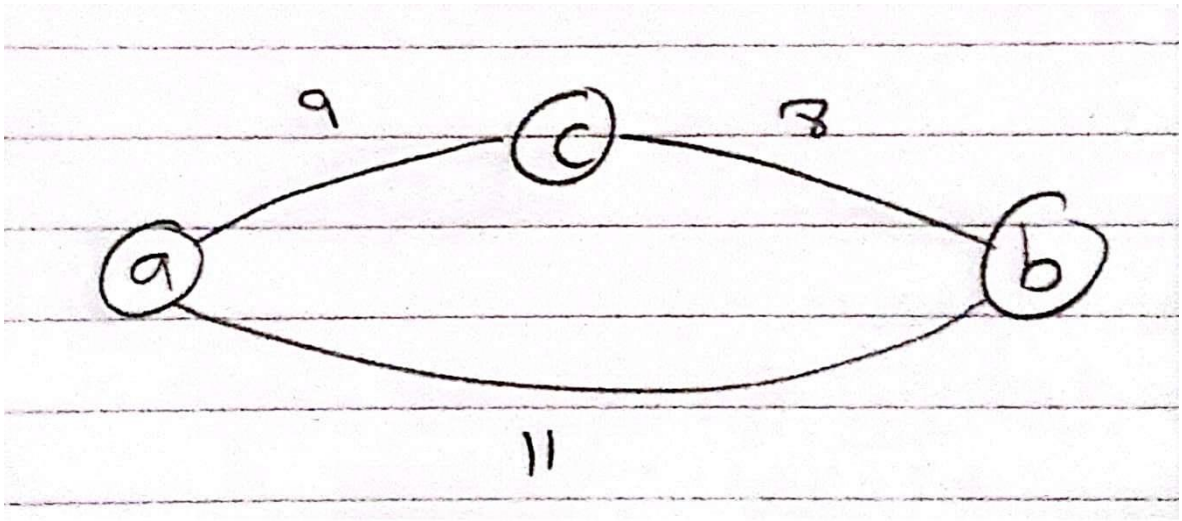
Perform Dijkstra on G:

S leaves the queue first and performs relax (S,T) and relax (S,V), leaving T.d = 9 and V.d=6. Then V leaves the queue and relax (V,U) is performed, leaving U.d = 9. Now U is extracted from the queue, however U.d is not the only vertex with minimal distance, as U.d=9, but T.d also equals 9.

- Let $G = (V, E)$ be an undirected connected graph, with positive weight function w over the edges. Assume that for two vertices $a, b \in V$, there exists a path between a and b that contains only edges e such that $w(e) < 10$.

  Claim: There exists a shortest path between a and b in G that consists only of edges e such that $w(e) < 10$.

False, counterexample:

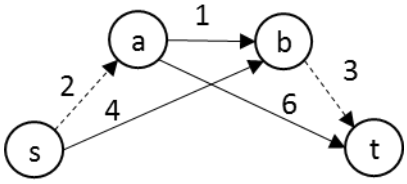


Problem 2 (24 pts.)

Let G=(V,E) be a directed graph with positive edge weights. The graph represents a network of roads and the weights denote the roads' lengths. Given is a source vertex $s \in V$ and a target vertex $t \in V$. As part of road improvement for an upcoming race, Mario is planning to upgrade some roads, and make driving on them take half as long. The set of roads that can be upgraded are given by the edge set $H \subseteq E$. Suggest an algorithm that decides which road (only one) $e \in H$, should be upgraded, so that the shortest path from s to t is as short as possible.

Example: In the given graph, for H={(s,a),(b,t)}, if (s,a) is selected to be upgraded, then the shortest path from s to t would be 5. If (b,t) is selected, then the shortest path would be 4.5. It is therefore best to upgrade (b,t).

Describe the algorithm formally, explain in several lines its correctness (state the main claim in the proof, but no need to prove it), state and justify the time complexity.



Algorithm:

Build a graph G' = (V' , E') s.t for all v ∈ V we make 2 copies, v0 and v1 in G' (i.e v0,v1 ∈ V'). For every edge (u,v) in E that cannot be upgraded make 2 copies (u0,v0) and (u1,v1). For edges that can be upgraded, make 3 copies (u0,v0) , (u1,v1) and (u0, v1).

Set the weights of these edges as follows: w((u0,v0)) = w((u1,v1)) = w(u,v)

W(u0,v1) = (0.5)w(u,v)

Now perform Dijkstra on G' from s0 to t1. Once Dijkstra is complete, iterate through the shortest path that was found using the predecessor variables (v.π) until you find an edge that is an element of H. This is the edge that must be returned.

Correctness:

There exists a shortest path of length d from s -> t in G, where the weight of an edge from H is halved if and only if there exists a shortest path from s0 -> t1 of length d in G'.

Time complexity:

Building the graph takes O(V' + E') = O(2V + (2E + |H|)) = O(V + E)

Dijkstra's algorithm is run once on G' takes time

O((V' + E') log(V')) <= O((2V + (2E + |H|)) log(2V)) <= O((2V + 3E) log(2V)) =

O((V+E)log(V))

To go through the shortest path and return the desired edge it takes O(E) time at most

Therefore total time is O((V+E)log(V))

Problem 3 (20 pts.)

Let $G = (V, E)$ be a directed strongly connected graph, w a weight function on the edges of G with $w(e) \geq 0$ for all $v \in V$. Let $V = \{v_1, \ldots, v_n\}$. Suggest an efficient algorithm that finds for every pair of nodes v_i, v_j, the length of the shortest path from v_i to v_j that passes through v_1. The output of the algorithm is a matrix D of size n^2, where the required value for v_i, v_j appears in $D[i][j]$. Describe the algorithm, justify its correctness in a few lines, specify and justify its time complexity.

Remarks: 1. If i=1 or j=1, the path need not pass in v_1 more than once.
2. A solution whose time complexity is $\Omega(n^3)$ will be granted 5 points.

Algorithm:

Build an n by n matrix D.

Build a graph G' = (V' , E') s.t for all v ∈ V we make 2 copies, v and v' in G' (i.e v0,v1 ∈ V'). For every edge (u,v) in E make 2 copies (u',v') and (u",v") where their weights is equal to the weight of (u,v). Now add an the edge (v1',v1") and set its weight to 0.

For every vertex Vi' (i ∈ {1,2,...,n}):

Perform Dijkstra on it. Store Vj".d (j ∈ {1,...,n}) in D[i][j].

Note that if such a path for Vi', Vj" does not exist in the graph then the value in D[i][j] will be infinity/null as that is what Dijkstra will return for Vj".d when performed on Vi' as a source.

Correctness:

A shortest path of length c from Vi -> Vj, that passes through V1 exists if and only if D[i][j] has value c stored in it.

Time complexity:

Building the graph takes O(V' + E') = O(2n + 2m + 1) = O(n + m)

Building the matrix takes $O(n^2)$

Dijkstra's algorithm on G' and storing the values in D takes time

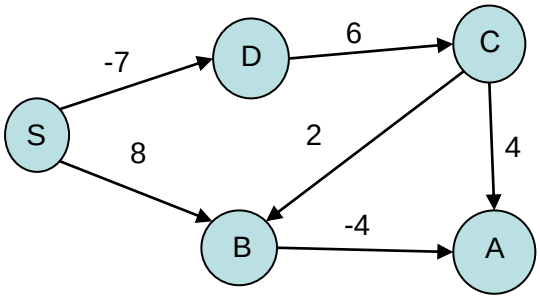
O((V' + E') log(V')) <= O((2n + (2m + 1)) log(2n)) <= O((2n + 2m) log(2n)) =

O((n+m)log(n))

Therefore total time is O((n+m)log(n))

Problem 4 (20 pts.)

1. Run Bellman-Ford algorithm on the following graph. Use the following edge order: SB, CA, CB, DC, SD, BA. Show in a table the values of v.d after every sweep.



	Sweep 1	Sweep 2	Sweep 3	Sweep 4
S.d	0	0	0	0
A.d	4	4	3	3
B.d	8	8	1	1
C.d	--	-1	-1	-1
D.d	-7	-7	-7	-7

2. Suggest an order of the edges such that all shortest paths are detected after

SD, SB, DC, BA, CB, CA

one sweep.

3. Suggest an order of the edges such that all shortest paths are detected after

SD, CB, DC, SB, BA, CA

two sweeps.

Problem 5 (20 pts.)

Run Floyd–Warshall algorithm on the graph given below by δ_0. Describe δ_i for each step i of the algorithm by filling the tables (no other explanations needed).

δ_0	1	2	3	4	5
1	0	6	4	∞	2
2	∞	0	∞	-4	3
3	∞	4	0	∞	7
4	∞	∞	1	0	7
5	1	−2	∞	2	0

δ_1	1	2	3	4	5
1	0	6	4	∞	2
2	∞	0	∞	-4	3
3	∞	4	0	∞	7
4	∞	∞	1	0	7
5	1	−2	5	2	0

δ_2	1	2	3	4	5
1	0	6	4	2	2
2	∞	0	∞	-4	3
3	∞	4	0	0	7
4	∞	∞	1	0	7
5	1	−2	5	-6	0

δ_3	1	2	3	4	5
1	0	6	4	2	2
2	∞	0	∞	-4	3
3	∞	4	0	0	7
4	∞	5	1	0	7
5	1	−2	5	-6	0

δ_4	1	2	3	4	5
1	0	6	3	2	2
2	∞	0	-3	-4	3
3	∞	4	0	0	7
4	∞	5	1	0	7
5	1	−2	-5	-6	0

δ_5	1	2	3	4	5
1	0	0	3	-4	2
2	4	0	-3	-4	3
3	8	4	0	0	7
4	8	5	1	0	7
5	1	−2	-5	-6	0

Solution:

δ_0	1	2	3	4	5
1					
2					
3					
4					
5					

δ_1	1	2	3	4	5
1					
2					
3					
4					
5					

δ_2	1	2	3	4	5
1					
2					
3					
4					
5					

δ_3	1	2	3	4	5
1					
2					
3					
4					
5					

δ_4	1	2	3	4	5
1					
2					

3					
4					
5					

δ_5	**1**	**2**	**3**	**4**	**5**
1					
2					
3					
4					
5					