Algorithms – Final Exam

1.2.20 – term A - Corona Edition

Prof. Tami Tamir

I.D:_____

Notebook number:_____

Question	Score
1	/19
2	/18
3	/20
4	/12
5	/18
6	/13
Total	/100

- Exam length: 3 hours for solving + 30 minutes for technical issues
- Any material written/files prepared in advance is allowed.
- You can cite and rely (without providing a proof) on theorems and claims that were proven in class or in the homework.
- In every question or section marked by ♣, you can write "I don't know" and get one quarter of the score (rounded down).

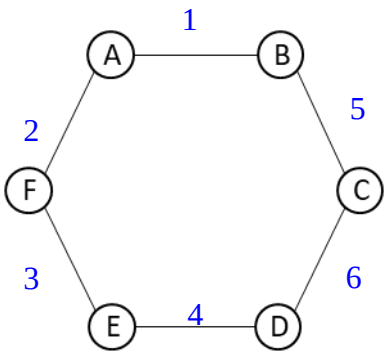
G O O D L U C K !

Question 1 (19 pts.)

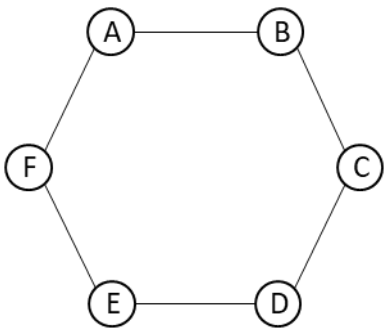
♣ **1.1 (3*3=9 pts.)** Consider the graph C_6 in the figure. Assume that Kruskal and Prim algorithms are performed on the graph in order to find a minimum spanning tree. The edge weights are known to be 1,2,3,4,5,6. That is, for every integer $1 \le i \le 6$, exactly one edge in the graph has weight i.

In each of the following sections (separately) write weights next to the edges, such that the described requirements are met, or briefly explain why there is no solution. If there is a solution, then there is no need to explain your answer.

a. In a run of Kruskal, as well as in a run of Prim that begins at node A, the edge (BC) is the last to join the MST.

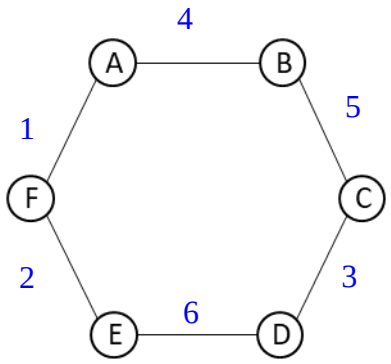


b. During a run of Kruskal, as well as during a run of Prim that begins at node A, the connected components are {A,B}, {C,D,E}, {F}.



No solution – In Prim's algorithm, the ST is built by growing a single component that includes the first node. All other components have size 1.

c. In a run of Kruskal, the edge (AB) is the fourth to join the MST. In a run of Prim that begins at node A, the edge (AB) is the third to join the MST.



♣ **1.2 (10 pts.)** Let G=(V,E) be a connected undirected graph with positive edge costs, c:E→R^+. Each of the edges is owned by an independent agent who offers the edge for sale. Given T, an MST of G, each of the agents who owns an edge in T receives the cost of the edge.

Let e∈T be an edge in the MST. Suggest the agent that owns e, an algorithm that returns whether and by how much he can increase the cost of e, such that even at the new cost, e will continue to be part of some MST of G. If there is no limit on the increase (e.g., if the graph is simply just the edge e), the algorithm should announce it. Formally, your algorithm should return the maximal x≥0, such that if the cost of e is c(e)+x then there exists an MST that includes e.

The time complexity of your algorithm should be O(|E|).
Describe the algorithm, analyze and justify the time complexity**. No need to justify the correctness of the algorithm.**

Algorithm: Remove e from T. Let T_1,T_2 be the resulting connected components.
Let H be the set of edges in the cut induced by T_1 (edges that have one endpoint in T_1 and one endpoint in T_2).
If H={e} then announce "unlimited increase" (e is included in every MST independent of its cost).
Otherwise, let e' the cheapest edge in H-{e}, announce "e can be increased by c(e')-c(e)".

Time complexity:
Calculate T_1 and T_2 by BFS or DFS from e's endpoints. O(|E|).
Identify the cut edges: O(|V|+|E|)=O(|E)|.
Find e': O(|H|)= O(|E|).

Correctness idea (not required, only for the next generations): By the blue rule if an edge is cheapest in a cut, then there exists an MST that includes it. The algorithm finds the maximal cost of e that will keep it cheapest in the cut.

Question 2 (18 pts.)

A student who has not attended lectures all semester plans to see their recordings in preparation for the exam. The course includes n lectures, each lasting 2 hours. The viewing software allows watching each lecture at normal speed or at double speed. For each lecture $1 \le i \le n$, we are given two positive values:

$V_2(i)$ - The benefit from watching lecture i at normal speed (in 2 hours).

$V_1(i)$ - The benefit from watching lecture i at double speed (in 1 hour).

Assumptions and constraints:

- One cannot see parts of lectures, or watch at other speeds (partial viewing has benefit 0).
- There is no dependence between different lectures, and for all i, $V_1(i) \le V_2(i)$.
- One can skip a lecture and not see it at all, in which case the benefit is 0, **but it is not possible to skip two consequent lectures.** For this constraint, there is no difference between watching at regular or double speed.

The student can devote at most H hours to the entire viewing. It is known that $n/2 < H < 2n$. In this question you will help the student to calculate, based on **dynamic programming,** what is the maximum benefit she can accumulate.

For every $0 \le i \le n$, $0 \le h \le H$, denote by B[i,h] the benefit the student can gain from the first i lectures, if they are allocated (together) h hours, **and lecture i is viewed** (at normal or double speed).

2.1 (2 pts.) What is the expression describing the solution's value (no need to explain)?

Max (B[n,H], B[n-1,H])

♣ **2.2** (12 pts.) Write a dynamic programming formula for calculating B[i, h]. Distinguish between the base and the general case. Explain the formula in words.

Base case: for all $1 \le i \le n$, $B[i,0]=0$, for all $0 \le h \le H$, $B[0,h]=0$.

In order to avoid reference to non-valid entries, either define $B[i,h]=0$ if $i \le 0$ or $h \le 0$ (this includes also the above base cases), or add $B[1,1]=V_1(1)$ and $B[1,2]=V_2(1)$ to the base cases and fill the table only if $1 \le i/2 \le h \le H$.

Recursive formula: for all $1 \le i \le n$, $i/2 \le h \le H$

$$B[i,h]= \max\{B[i-1,h-1]+V_1(i),\ B[i-1,h-2]+V_2(i),\ B[i-2,h-1]+V_1(i),\ B[i-2,h-2]+V_2(i)\}.$$

Explanation: For every $j \in \{1,2\}$, $h-j \ge 0$, if the student spends j hours watching lecture i it gains benefit $V_j(i)$. If lecture i-1 is viewed, then the remaining h-j hours will produce benefit B[i-1,h-j]. If lecture i-1 is not viewed, then lecture i-2 must be viewed, and the remaining h-j hours will produce benefit B[i-2,h-j].

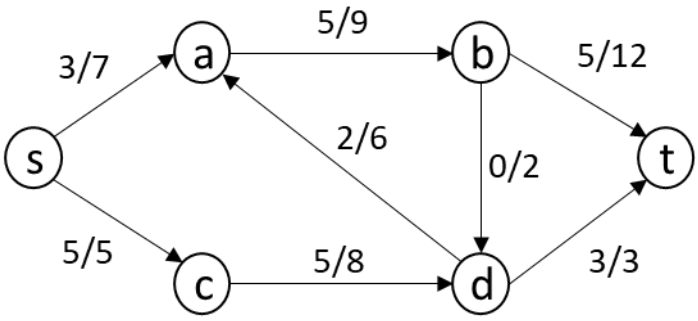
♣**2.3** (4 pts.) Describe the DP-table and the order it is filled. Analyze the time-complexity of the algorithm.

The table has dimensions (n+1)*(H+1). The base case is the first column – corresponding to m=0, and the first row – corresponding to i=0. For each entry in row i, we need four entries from row i-1. The table can be filled row after row. Each entry takes O(1), all together O(nH).

Question 3 (20 pts.)

♣ **3.1 (10 pts.)** The figure presents a flow network with a legal flow. Every edge e∈E is labeled by f(e)/c(e) . It is known that the given flow is a result of **two** iterations of Ford & Fulkerson algorithm.

No need to justify your answers in this section.



1. What was the augmenting path in the first iteration?

   s-c-d-a-b-t

2. What was the augmenting path in the second iteration?

   s-a-d-t

3. Suggest an augmenting path for the next (third) iteration:

   s-a-b-t

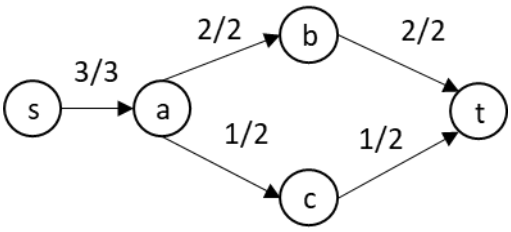
4. What is the value of the maximum flow in the network? 12

5. Write all sets of nodes that define a min-cut in the network:

   {s}, {s,a,c,d}

♣ **3.2 (10 pts.)** Given is a flow network: a directed graph $G = (V, E)$, source and target nodes $s, t \in V$, and a capacity function $c: E \to \mathbb{N}$. Also given is a maximum flow function $f: E \to \mathbb{N}$. Suggest an algorithm whose time complexity is $O(|E| + |V|)$ for deciding whether there exists an additional maximum flow function $f'E \to \mathbb{N}$, which is different from f (but has the same value). If the answer is positive, the algorithm should suggest such a function.

Example: For the following input, the algorithm should output that an additional flow function of value 3 exists: $f'(s,a) = 3,\ f'(a,b) = 1,$ $f'(a,c) = 2,\ f'(b,t) = 1, f'(c,t) = 2$



a. Complete the algorithm (the first line is given), and justify shortly its time complexity.

1. Let $R_f = (V, E')$ be the residual network corresponding to the maximum flow f.

1. Run DFS to check whether R_f includes a directed cycle.

2. If R_f does not include a cycle, then there is no additional max-flow

3. If R_f includes a cycle, let C be a simple directed cycle in R_f. Let δ be the minimal capacity of an edge in C (in R_f), for every forward edge in C, let f"(e)=f(e)+ δ. For every backward edge in C, let f"(e)=f(e)- δ. For every other edge, let f"(e)=f(e).

Time complexity: Building R_f, Running DFS, retrieve the cycle, define f' – all are O(|E|).

Remark: any integer in the range[1,.., δ] can be selected as the value of the change along the cycle.

b. Justify shortly: If the algorithm decides that the answer is positive, and outputs a flow function $f': E \to \mathbb{N}$, then f' is a legal max-flow. **No need** to justify the other direction (if the answer is negative, then f is a unique max-flow function).

The edge condition is preserved by the choice of δ.
The vertex condition is preserved since the flow is updated along a cycle. The proof is identical to the validity of the flow after each iteration in F&F algorithm.
The value of the flow does not change: If s is not part of the cycle, then the flow on edges leaving s remains the same, if s is part of the cycle, then since one edge in the cycle enters s and one leaves s, we only change the way the flow that leaves s is distributed.

♣ Question 4 (12 pts.)

In a faraway kingdom there are n districts. For all $1 \le j \le n$, d_j residents live in district j. Following an epidemic that broke out in the kingdom, it was decided to vaccinate the residents. King officials have reached an agreement with a company that produces the vaccines. According to the agreement, n packages of vaccines will arrive in the kingdom. For every $1 \le i \le n$, package i includes $p_i \ge 0$ shots. Due to logistical restrictions, it was decided that every district would receive one package of vaccines, and only residents of this district will be vaccinated with shots from this package. If in a particular district the number of residents is lower than the number of shots in the package, then the excess shots will be discarded and will not be used. If the number of residents is higher than the number of shots in the package, then only some of the residents will be vaccinated. The king instructed his advisers to decide how to allocate the packages to the districts so that the total number of residents to be vaccinated throughout the kingdom would be maximal.

a. (5 pts.) One of the king's advisers proposed the following algorithm:
Find a pair ⟨package i, district j ⟩ such that $|p_i - d_j|$ is minimal. Allocate package i to district j, and proceed in the same way with the remaining packages and districts.

Show by example that the algorithm is not optimal. Describe the input, the output of the algorithm, and the optimal solution.

Let n=2, d1=6, d2=3, p1=4, p2=1.
The algorithm will match (d2,p1) (d1,p2) and will achieve min(3,4)+min(6,1)=3+1=4 vaccinated residents. The matching (d1,p1), (d2,p2) achieves min(6,4)+min(3,1)=4+1=5 vaccinated residents.

2. (5 pts.) Suggest an optimal greedy algorithm for the problem. No need to specify or analyze the time complexity.

Sort the districts $d_1 \ge d_2 \ge \ldots \ge d_n$. Sort the packages $p_1 \ge p_2 \ge \ldots \ge p_n$.
For i=1..n, allocate package i to district i.

3. (3 pts.) State the corresponding greedy choice property. No need to prove it.

There exists an optimal solution in which the largest package is allocated to the most populated district.



♣ Question 5 (18 pts.)

Given is a directed graph G = (V, E) with positive edge weights $w:E \rightarrow R^+$. The graph represents a network of ants' burrows, and the weight on each edge indicates the length of the burrow in centimeters. A sesame seed is placed in node $t \in V$. Given is a set of nodes $A \subseteq V$, and a function $r:A \rightarrow R^+$. In each $v \in A$ there is a single ant. All the ants depart simultaneously and proceed towards the sesame seed. Every ant proceeds in a shortest path from its origin node to t. The ant that departs from node v proceeds at a constant speed r(v), i.e., for every $e \in E$, it takes this ant w(e)/r(v) time-units to cross burrow e.
Suggest an efficient algorithm that returns a vertex $v \in A$ such that the ant departing from v reaches the sesame seed first (there may be that several ants will arrive first together, in which case any one of them can be returned).
Describe the algorithm in words and formally, prove its correctness (state the main claim, no need to prove it), state and justify the time complexity.

Algorithm:
1. Given G, Build the following graph G'=(V,E')
V' = V, $E'=\{(v,u)|(u,v) \in E\}$, that is reverse the edge directions.
For all e, w'(v,u)=w(u,v).
2. Run Dijkstra G' with t as a source.
3. Let $v \in A$ be a node in A for which d(t,v)/r(v) is minimal. Return v.

Description: Reverse the edges of G, run Dijkstra from t. For every ant, divide the length of the shortest path by its speed, in order to get the time to reach the sesame seed.
Correctness: For every $v \in A$, G includes a path on which the ant that leaves v can reach the sesame seed within u time units iff G' includes a path from t to v of length u*r(v).

Time complexity:
|V'|=|V| and |E'|=|E|.
Therefore, building G' takes O(|V|+|E|).
Running Dijkstra takes $O((|V'|+|E'|)\log|E'|)= O((|V|+|E|)\log|E|)$
Calculating the minimum: $O(|A|) \leq O(|V|)$.

All together: $O((|V|+|E|)\log|E|)$

Question 6 (13 pts.)

♣ **6.1** (4 pts) Given is an undirected bipartite, $G=(V_1 \cup V_2, E)$, $E \subseteq V_1 \times V_2$, $E \neq \emptyset$.
It is known that all the vertices in the graph have the same degree.
Prove or refute: $|V_1|=|V_2|$.

Proof: Let r be the degree of all the vertices. We know that every edge has one endpoint in each side, therefore $|E|=|V_1|r$ and also $E=|V_2|r$, implying $|V_1|=|V_2|$.

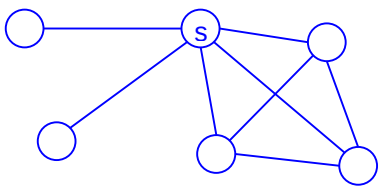
♣ **6.2** (9 pts) Let G be a simple connected undirected graph over n nodes. Let $s \in V$. DFS(s) is performed. In the resulting undirected spanning tree, the degree of s is d.
Complete the following statement with the minimal and maximal possible number of edges in G.
Your answer should be a function of n,d. Explain your answer (each side separately).

n-1	$\leq \lvert E \rvert \leq$	(d-1)+(n-d)(n-d+1)/2

Lower bound: G is connected, so it must include at least n-1 edges. If G is a tree in which s has degree d, then the condition is fulfilled and $|E|=n-1$.

Upper bound: s has degree d, and there are no edges between the sub-graphs connected to each of the d neighbors (as otherwise, they will not be in different subtrees of the DFS). Let $(n_1,n_2,\ldots,n_d)$ be the number of vertices, except s, in each subtree.
The maximal number of edges in subgraph i is $n_i(n_i+1)/2$ – if, together with s, it is a complete graph.
We know that $n_1+n_2+\ldots+n_d=n-1$. In order to maximize $\Sigma i\ n_i(n_i+1)/2$, we want to maximize the size of the largest subgraph (since x(x+1) is a convex function), and this is achieved by setting $(n_1,n_2,\ldots,n_d) = (n-d,1,1,\ldots,1)$. We have (n-d)(n-d-1)/2 edges in the large subgraph (complete graph over n-d+1 vertices), and one edge in each of the other d-1 subgraphs (can be viewed as complete graphs over 2 vertices).

Example for n=6, d=3:



Remark: Almost full credit was given to correct upper bound without a complete mathematical justification.