NAME: ______________________

STUDENT ID: ______________________

CSE 421
Introduction to Algorithms
Final Exam Spring 2019

DIRECTIONS:

- Answer the problems on the exam paper.
- Justify all answers with proofs (except for Problem 1), unless the facts you need have been stated or proven in class, or in Homework, or in sample-midterm/final.
- If you need extra space use the back of a page or two additional pages at the end
- You have 100 minutes to complete the exam.
- Please do not turn the exam over until you are instructed to do so.
- Good Luck!

1	/40
2	/25
3	/20
4	/25
5	/25
6	/25
Total	/165

1. (40 points, 4 each) For each of the following problems answer **True** or **False** (no proof needed).

   (a) The minimum vertex cover in a complete graph with n vertices has size n.

   (b) For any pair of NP-hard problems A, B we have $A \leq_p B$.

   (c) For any pair of NP-Complete problems A, B we have $A \leq_p B$.

   (d) For every weighted graph G (with possibly *negative* edge weights) Kruskal's algorithm finds the minimum spanning tree.

   (e) Let G be a directed graph, f be a s-t flow, and let (A, B) be a s-t cut. Then,

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \leq cap(A, B).$$

(f) If the capacity of every edge in a directed graph is increased by 1 then the value of the maximum flow will be increased by exactly 1.

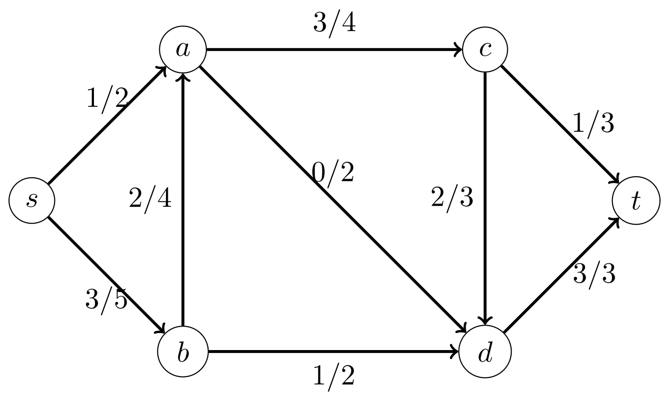
(g) If $P \neq NP$ then for every problem $A \in NP$ and $B \in P$ we have $A \not\leq_p B$, i.e., there is no polynomial time reduction from A to B.

(h) If the size of the maximum clique can be found in time $2^{O(\sqrt{n})}$ then the size of the minimum vertex cover can be found in time $2^{O(\sqrt{n})}$.

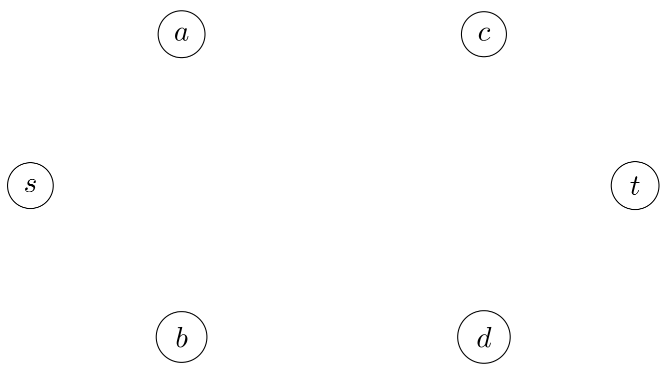
(i) There is a linear time algorithm for finding the $n/3 + 1$'th largest number in a list of n numbers

(j) Let G be an undirected graph with n vertices and $n - 1$ edges then for every pair of vertices, u, v, there is *exactly one* path from u to v in G.

2. (20 points) Consider the following flow network with a flow f shown. An edge labelled "b" means that it has capacity b and flow 0. An edge labelled "a/b" means that the flow on that edge is a and the capacity is b.

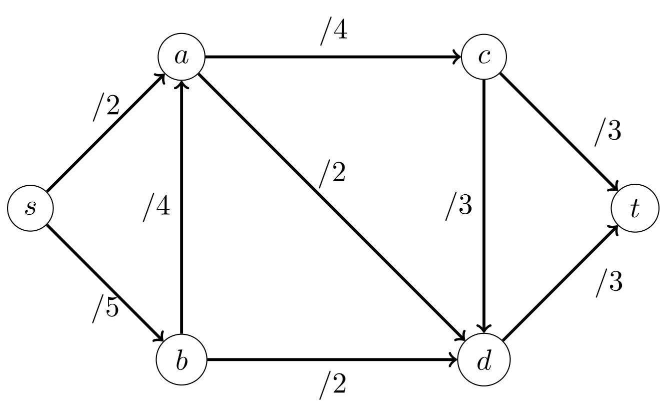


(a) (10 points) Draw the residual graph G_f.

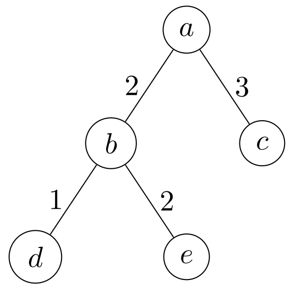


(b) (5 points) What augmenting path in this graph would result in the greatest increase in flow value? (List the names of the vertices on this path in order.)

(c) (10 points) On the diagram below, indicate the new flow values resulting from augmenting along the path you found in part (b).



3. (20 points) Given a tree T where every edge e has weight w_e. Design a polynomial time algorithm that outputs the weight of the maximum weight matching of T. Your algorithm should just output the weight of the maximum matching. For example, in the following tree the maximum matching matches a to c and b to e and it has weight $2 + 3 = 5$.



4. (20 points) A sequence of numbers $a_1, \ldots, a_m$ is called *mirrored* if m is even and for all $1 \leq i \leq m$, $a_i = a_{m-i+1}$. Given a sequence of numbers $x_1, \ldots, x_n$ we want to determine the smallest number of new numbers to be inserted into the sequence in order to obtain a mirrored sequence. For example, for the input $3, 2, 1, 2, 1, 2, 2$ the following

$$3, 2, \underset{\uparrow}{2}, 1, 2, \underset{\uparrow}{2}, 1, 2, 2, \underset{\uparrow}{3}$$

is the smallest mirrored sequence that can be obtained by adding 3 new numbers marked as above. Your code should run in time polynomial in n and output the number of new numbers that you add, e.g., 3 in the above example.

5. (25 points) For a pair of nodes u, v in a graph $G = (V, E)$ we write $dist(u, v)$ to denote the length of the shortest path from u to v in G. Recall that the diameter of G is the maximum distance between any pair of nodes, i.e., $max_{u,v} dist(u, v)$. For example, in HW2-P3 we designed a linear time algorithm which finds the diameter of a tree. Given a connected undirected *unweighted* graph $G = (V, E)$, design a 2 approximation linear time algorithm for the diameter of G. Your algorithm must run in time $O(m + n)$, and it should output an integer which is a two approximation of diameter of G. Here m is the number of edges and n is the number of vertices.

   Hint: Recall that for any three vertices u, v, w, $dist(u, v) \leq dist(u, w) + dist(w, v)$.

6. (25 points) The *Teacher Assignment* problem is: given a set of teachers $T = \{t_1, \ldots, t_n\}$ and a set of courses $C = \{c_1, \ldots, c_m\}$ determine an assignment of teachers to courses. The input to the problem has a bipartite graph $G = (T, C, E)$ where an edge (t, c) indicates that teacher t can teach class c. For each teacher t_i there is an integer u_i giving the number of courses that t_i must teach, and for each course c_j there is an integer d_j indicating how many teachers must be assigned to the course. A teacher t can be assigned at most once to course c (in other words, if multiple teachers are required for a course, they must be distinct). Design an algorithm that run in time polynomial in m, n and $t_1, \ldots, t_n$ and $c_1, \ldots, c_m$ and outputs "yes" if such a assignment exists and "no" otherwise.