NAME: ______________________________

CSE 421
Introduction to Algorithms
Sample Final Exam 2023

Anup Rao

June 1

DIRECTIONS:

- Answer the problems on the exam paper.
- You are allowed a single cheat sheet.
- Justify all answers with proofs, unless the facts you need have been proved in class or in the book.
- If you need extra space use the back of a page
- You have 1 hour and 50 minutes to complete the exam.
- Please do not turn the exam over until you are instructed to do so.
- Good Luck!

1	/60
2	/20
3	/20
4	/20
5	/25
Total	/130
Extra	/ 10

1. (60 points, 5 each) For each of the following problems answer **True** or **False** and BRIEFLY JUSTIFY you answer.

   (a) Let (S, T) be a minimum st-cut in a flow graph (G, s, t). If the capacity of every edge in G is increased by 2 then the value of the maximum flow in G is increased by 2 times the number of edges from S to T in G.

   Solution: False. There might be another min-cut with fewer edges, and its capacity will be increased by less.

   (b) There is an exponential time algorithm for $3SAT$.

   Solution: True, NP is contained in EXP.

   (c) There is a linear time algorithm for finding the $n/10$'th largest number in a list of n numbers.

   Solution: True, we saw this in class.

   (d) If $T(n) = 3T(n/2) + n$ for $n \geq 2$ then $T(n)$ is $\Theta(n^{\log_3 2})$.

   Solution: False, the master theorem says $O(n^{\log_2 3})$.

   (e) If $T(n) = 12T(n/4) + n^2$ for $n \geq 4$ then $T(n)$ is $\Theta(n^2)$.

   Solution: True, by master theorem.

(f) If $3SAT$ has a linear time algorithm, then so does every problem in NP.
Solution: False, the reductions can be polynomial time, so we only get that NP is contained in P.

(g) Every decision problem has an exponential time algorithm.
Solution: False, not the halting problem.

(h) The average degree of a vertex in a tree with n vertices is exactly 2.
Solution: False, it is $2(n-1)/n = 2 - 1/n$.

(i) If e is an edge whose weight is lower than the weight of all other edges in an undirected graph, then every minimum spanning tree must contain e.
Solution: True, by the cut property.

(j) In a flow network, the value of any valid flow is at most the capacity of any $s - t$ cut.
Solution: True, we proved this.

(k) There is a polynomial time algorithm for Vertex Cover that finds the optimal vertex cover up to a factor of 2.
Solution: True, we gave a greedy algorithm.

(l) If the optimal solution to a linear program is 10, then the corresponding dual linear program must have a solution of finite value.

Solution: True, and that value must be 10 by strong duality.

2. (20 points) **Assigning teachers to courses**

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).

Describe how network flow can be used to find an assignment of teachers to courses. If no assignment is possible that meets these constraints, the algorithm should report failure. Be sure to argue the correctness of your solution.

Solution:

Make a flow netweork with vertices $s, t_1, \ldots, t_n, c_1, \ldots, c_m, t$. There are three types of edges: edges of the type (s, t_i) with capacity t_i, edges of the type (t_i, c_j) with capacity 1, and edges of the (c_j, t) with capacity d_j.

Find the maximum flow in this network using the capacity scaling algorithm. By the integrality theorem, the max flow will have a solution where all flows are integers.

If the flow fills all the edges from s and into t to capacity, then the problem can be solved. Morever, the edges (t_i, c_j) carrying flow 1 in the final solution corresponds to the teacher assignments.

To prove the correctness, note that every possible solution to the teacher assignment corresponds to a flow that fills all the edges from s and into t, and such a flow must be equal to a maximum flow because its value is equal to the capacity of the cut that separates s from the rest of the flow network.

(You can also use space on the next page for your answer)

3. (20 points) Let $G = (V, E)$ be an undirected graph. Suppose that each edge e has a cost $c(e)$, with $c(e) \in \{1, 2, 3\}$. Describe an $O(n+m)$ time algorithm to compute a minimum spanning tree for G. HINT: Observe that if all the edge costs were exactly the same, then *any* spanning tree would be a minimum spanning tree of the graph.

   Solution:

   (a) Let G_1 denote the graph using only the edges of weight 1. Use the BFS algorithm to find a spanning tree for each connected component of G_1. Let T be the set of edges found in this process.

   (b) Next, let G_2 be the graph whose vertices are the connected components of T. There is an edge $\{u, v\}$ in G_2 if and only if G has an edge of weight 2 from connected component u to connected component v in T. Find a spanning tree for each connected component of G_2. Add the set of weight 2 edges used in this step to T.

   (c) Finally, let G_3 be the graph whose vertices are the connected components of T. There is an edge $\{u, v\}$ in G_3 if and only if there is an edge of weight 3 from connected component u to connected component v in G. Find a spanning tree for each connected component of G_3. Add the set of weight 3 edges used in this step to T.

   The algorithm runs in time $O(m+n)$. Each computation of spanning trees for the connected components can be done using BFS as shown in class. G_1, G_2, G_3 can also be computed in time $O(m+n)$ by scanning all the edges of G and checking whether or not they go between different connected components.

   To argue that the algorithm is correct, consider sorting the edges of the graph in increasing order of weights, such that if two edges have the same weight, and exactly one of them belongs to T, then the one that does not belong to T comes later in the sorted order.

   We claim that Kruskal's algorithm will output T when give the edges in this sorted order. This is because the edges of T never create a cycle, and the edges that do not belong to T must always go within a connected component found in the above algorithm, and so will always create a cycle.

4. (20 points) Consider the linear program:

$$\begin{aligned} &\text{maximize } x_1 - 2x_3 \\ &\text{subject to} \\ &x_1 - x_2 \leq 1 \\ &2x_2 - x_3 \leq 1 \\ &x_1, x_2, x_3 \geq 0 \end{aligned}$$

Prove that $(x_1, x_2, x_3) = (3/2, 1/2, 0)$ is an optimal solution.

Solution: The given solution has value 3/2.

The dual of the program is:

$$\begin{aligned} &\text{minimize } y_1 + y_2 \\ &\text{subject to} \\ &y_1 \geq 1 \\ &-y_1 + 2y_2 \geq 0 \\ &-y_2 \geq -2 \\ &y_1, y_2 \geq 0 \end{aligned}$$

Now we see that setting $y_1 = 1, y_2 = 1/2$ gives a solution to the dual of value 3/2. By weak duality, both of these solutions must be optimal.

5. (25 points) The following problem can be useful for data compression. The input is a string y of length n and a list of k strings $x_1, \ldots, x_k$ of lengths $m_1, \ldots, m_k$ respectively, where each x_i is represented as an array $x_i[1] \cdots x_i[m_i]$ and y is $y[1] \cdots y[n]$. The problem is to determine the smallest number of copies of strings from $x_1, \ldots, x_k$ that can be concatenated together to produce y. For example, if $x_1 = a$, $x_2 = ba$, $x_3 = abab$, and $x_4 = b$ and $y = bababbaababa$ then we can write $y = x_4 x_3 x_2 x_3 x_1$ which is optimal, so the answer is 5.

   (a) (10 points) For $i \geq 0$, let $Opt(i)$ be the optimal number of strings required to produce the string $y[1] \cdots y[i]$. Describe a recursive algorithm for computing $Opt(n)$. Don't forget the base case.

   Solution: We can express $Opt(i)$ in terms of $Opt(j)$, with $j < i$, as follows.

   If in the optimal solution the final part of y is generated using x_r, then we must have $y = \ldots x_r$, so $Opt(i) = 1 + Opt(i - m_r)$. Thus, we get

$$Opt(i) = \min_{r: x_r \text{ is a suffix of y}} 1 + Opt(i - m_r)$$

   For the base case, we have $Opt(0) = 0$.

   (b) (10 points) Describe how you can compute $Opt(n)$ efficiently using an iterative dynamic programming algorithm.

   Solution:

   i. Set $M(0) = 0$.

   ii. For $i = 1$ to n, compute $M(i)$ using the above formula.

$$M(i) = \min_{r: x_r \text{ is a suffix of y}} 1 + M(i - m_r)$$

   (c) (5 points) What is the running time of your algorithm in terms of n, k and $m = \max_i m_i$?

   Solution:

   The running time is at most $n \sum_r m_r \leq O(nkm)$, since for each i we need to check to find which of the strings are valid suffixes.

(You can also use space on the next page for your answer)

6. (10 points Extra Credit) Give an $O(n^3)$ time algorithm for finding a 4-cycle in an undirected graph, if one exists. HINT: If a graph has a 4-cycle $abcd$, then there must be two distinct paths of length two between a and c. For every vertex b, keep track of the paths of length 2 that are generated with b in the middle.

   Solution:

   (a) Maintain a two dimensional array $A(i, j)$ where initially all of the entries are set to 0.

   (b) For every vertex of the graph b, for every pair of neighbors a, c of b, if $A(a, c) = 0$, set $A(a, c) = b$ otherwise, we have found a cycle $a, A(a, c), c, b$.

   The running time of the algorithm is $O(n^3)$.