CSC 226 PRACTICE FINAL EXAM

NAME: _____ STUDENT NO: _____

Instructor: Nishant Mehta Duration: 3 hours

NOTES:

1. THIS EXAM PAPER SHOULD HAVE 10 PAGES. COUNT THE NUMBER OF PAGES IN THIS EXAM PAPER BEFORE BEGINNING TO WRITE. REPORT ANY DISCREPANCY IMMEDIATELY TO THE INVIGILATOR.
2. THIS EXAM IS CLOSED BOOK; NO NOTES/BOOKS OR ELECTRONIC DEVICES ALLOWED.
3. ANSWER ALL QUESTIONS ON THE EXAM PAPER. USE THE BACK IF NEEDED.
4. SCRATCH PAPER IS AVAILABLE FROM THE INVIGILATORS.
5. THERE IS A TOTAL OF 110 MARKS.

Question	Potential Marks	Actual Marks
1	38	
2	4	
3	6	
4	8	
5	6	
6	8	
7	8	
8	6	
9	10	
10	5	
11	5	
12	6	
Total:	110	

Use the convention that the height of a tree is the length of the longest root-to-leaf path (so that a tree consisting only of a root node is of height 0).

1. [38 marks] Short answer questions (2 marks each)

   (a) Suppose that an algorithm has runtime $T(n)$ which satisfies the recurrence relation $T(n) = 3T(\lfloor n/3 \rfloor) + cn$ for a positive constant c, and for all $k \leq 5$ we have $T(k) \leq 2$. What is the asymptotic runtime of this algorithm?

   Answer: $\Theta(n \log n)$

   (b) What is the maximum number of nodes in a 2-3 tree of height k?

   Answer: $2^{k+1} - 1$

   (c) How many red links are there in a red-black tree of height 2 that has the maximum possible number of nodes?

   Answer: 0

   (d) What is the information-theoretic lower bound on the number of comparisons needed to sort an array of n distinct integers? You may use $\Omega(\cdot)$ notation.

   Answer: $\Omega(n \log n)$

   (e) In terms of how much code can be shared between the algorithms, which algorithm is most similar to Dijkstra's algorithm?

   A. Ford-Fulkerson
   B. Floyd-Warshall
   C. Edmonds-Karp
   D. Kruskal's algorithm
   E. Prim's algorithm

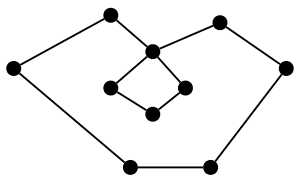
   (f) True or False: The Floyd-Warshall algorithm is a correct algorithm for the all-pairs shortest paths problem if the graph has negative weight edges, provided there are no negative cycles.

   Answer: True

   (g) Complete the characterization of Eulerian graphs: A graph is Eulerian if it has at most one nontrivial component and all its vertices have even degree.

   (h) A Hamiltonian path is a path which visits each vertex exactly once.

   (i) Is the graph below Hamiltonian?

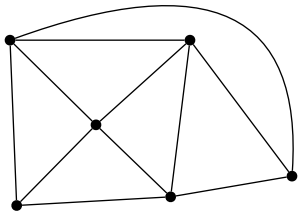


   Answer: No

(j) One implication of Kuratowski's theorem is that the complete bipartite graph $K_{x,y}$ is not planar. What are x and y?

Answer: $x = 3, y = 3$

(k) How many faces are in the graph below?



Answer: 7

(l) The Edmonds-Karp algorithm involves using the Ford-Fulkerson algorithm as well as which of the following subprocedures for finding an augmenting path?

A. Floyd-Warshall algorithm
B. Dijkstra's algorithm
C. Depth-first search
D. Breadth-first search

(m) Consider an s-t cut (A, B) in a flow network. We say that a directed edge is "out of A" if it originates in A and ends in B. Likewise, we say that a directed edge is "into A" if it originates in B and ends up in A. The value of a flow f is equal to

A. $\sum_{e \text{ out of } A} f(e)$
B. $\sum_{e \text{ into } A} f(e)$
C. $\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into of } A} f(e)$
D. $\sum_{e \text{ into } A} f(e) - \sum_{e \text{ out of } A} f(e)$

(n) What does the Max-Flow Min-Cut Theorem say?

Answer: The maximum value among all s-t flows is equal to the minimum capacity among all s-t cuts.

(o) Consider a flow network. Let f be an arbitrary flow with value $v(f)$, and let (A, B) be an arbitrary s-t cut with capacity $c(A, B)$. Which one of the following relationships is guaranteed to be true?

A. $c(A, B) \leq v(f)$
B. $c(A, B) = v(f)$
C. $c(A, B) \geq v(f)$

(p) Consider the following type of algorithm for finding a solution based on an input of size n. First, partition the input into 3 roughly equal pieces, yielding 3 disjoint subproblems. Run the algorithm recursively on each piece. Somehow combine the solutions for the subproblems to obtain a solution for the original problem. What algorithm design paradigm does this algorithm best fit into?

A. Divide and Conquer
B. Greedy
C. Dynamic Programming
D. Brute Force

(q) In what algorithm design paradigm is memoization used?

A. Divide and Conquer
B. Greedy
C. Dynamic Programming
D. Brute Force

(r) Consider the binary tree corresponding to an optimal prefix code for a given text. Where in the tree should the letter with the lowest frequency of occurrence be placed?

Answer: at the bottom of the tree

(s) What is the worst-case asymptotic runtime of Quickselect when using the median-of-medians pivot (itself obtained via recursive calls to Quickselect)?

Answer: $\Theta(n)$

(Above answer assumes that the groups of medians are of constant size 5 or larger. If the groups of medians are of size 3, the runtime is worse)

2. [4 marks] Let X be a Bernoulli random variable with success probability 0.5, and let Y be Bernoulli random variable with success probability 0.25. What is $\mathsf{E}[2X + 4Y]$?

Solution: First, observe that:

$\mathsf{E}[X] = 0.5 \cdot 1 + (1 - 0.5) \cdot 0 = 0.5$

$\mathsf{E}[Y] = 0.25 \cdot 1 + (1 - 0.25) \cdot 0 = 0.25$

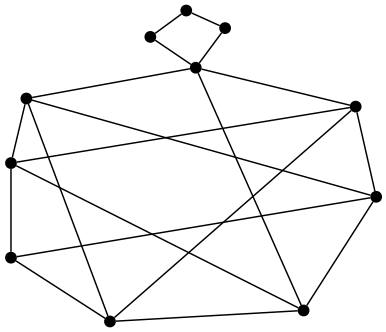
From linearity of expectation, we have $\mathsf{E}[2X + 4Y] = 2\,\mathsf{E}[X] + 4\,\mathsf{E}[Y]$, which is equal to $1 + 1 = 2$

3. [6 marks] Let G be a connected weighted graph with n vertices and which has non-negative edge weights. Describe a modification of the Bellman-Ford algorithm which identifies all single-source shortest paths of length k (for some fixed $k < n$) but which has a lower runtime than the full Bellman-Ford algorithm.

Solution: The intention was for the question to be for all shortest paths of length less than or equal to k. For this version, just run the outer loop of the Bellman-Ford algorithm k times instead of $n-1$ times. To see why this is correct, consider a shortest path from vertex s to a vertex v whose length is $m \leq k$. Let this shortest path be $v_0, v_1, \ldots, v_m$ for $v_0 = s$ and $v_m = v$. Then the Bellman-Ford algorithm with the aforementioned modification will ensure that the sequence of calls $\textsc{Relax}(v_0, v_1), \ldots, \textsc{Relax}(v_{m-1}, v_m)$ occurs, and so the shortest path will be discovered by this modified algorithm.

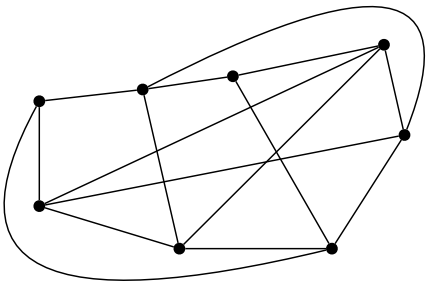
4. [8 marks] Two graphs are below. For each graph, indicate whether or not it is 2-colorable. If any graph is not 2-colorable, show one odd cycle in the graph.

First graph:



Solution: Yes it is 2-colorable.

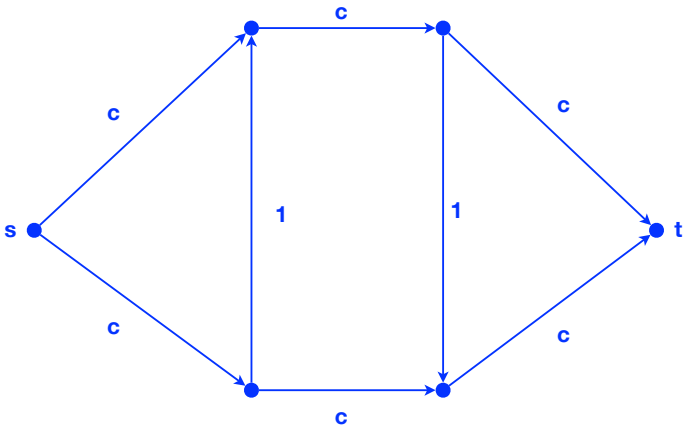
Second graph:



Solution: No, it is not 2-colorable. There is an odd cycle consisting of 3 edges (a triangle), using the top right vertex, the vertex below it that is slightly to the right, and the leftmost vertex that is directly connected to the each of the aforementioned 2 vertices via an edge.

5. [6 marks] In the worst case (and using poor choices for augmenting paths), how does the runtime of the Ford-Fulkerson algorithm scale with $\max_e c_e$, where the max is taken over all edges and c_e is the capacity of edge e? You may assume that the number of edges present in the original flow network is bounded by an absolute constant (say, 10).

Solution: Since the method for finding augmenting paths was not indicated, the runtime will be expressed in terms of the number of augmenting paths the algorithm might find before termination, in the worst case. In the worst case, the runtime of the algorithm can be as large as $K \cdot \max_e c_e$, where K is some positive constant depending on the number of edges present in the original flow network. To see how it can be as large as $2 \max_e c_e$, consider the below flow network with the specified capacities.

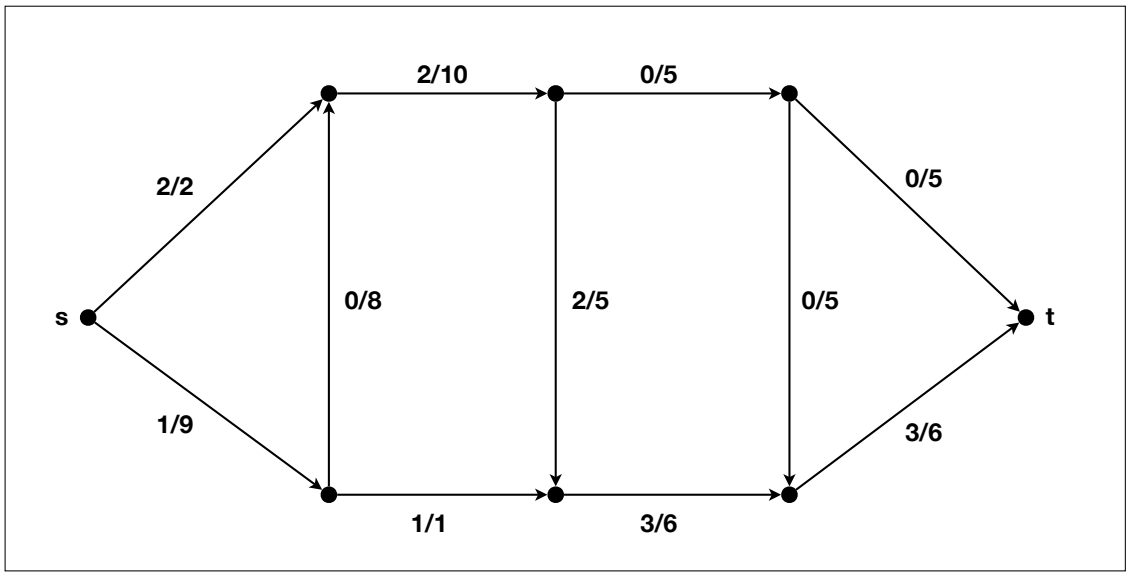


By ensuring that each augmenting path contains an edge of capacity 1, there can as many as $2c = 2 \max_e c_e$ augmenting paths before the maximum flow is found.

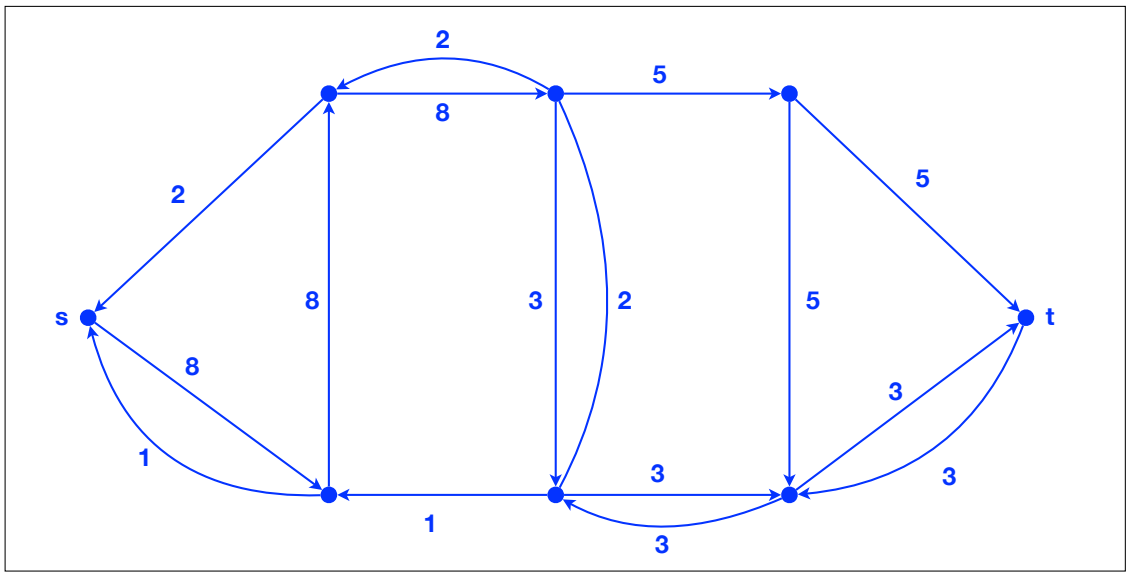
(To get full marks on this question, it would have been sufficient to explain how to force the algorithm to find $\max_e c_e$ augmenting paths. Increasing the number of augmenting paths beyond that, like with the factor of $K = 2$ as above, is not necessary.)

6. [8 marks] In the flow network below, the label x/y indicates that there currently is a flow of x units along the edge, and the capacity of the edge is y. For instance, 1 unit (out of a total possible of 9 units) is flowing along the left-most vertex s to the vertex that is below it immediately to the right. Given the flow shown below, draw the corresponding residual network. Be sure to indicate the capacities for each edge.

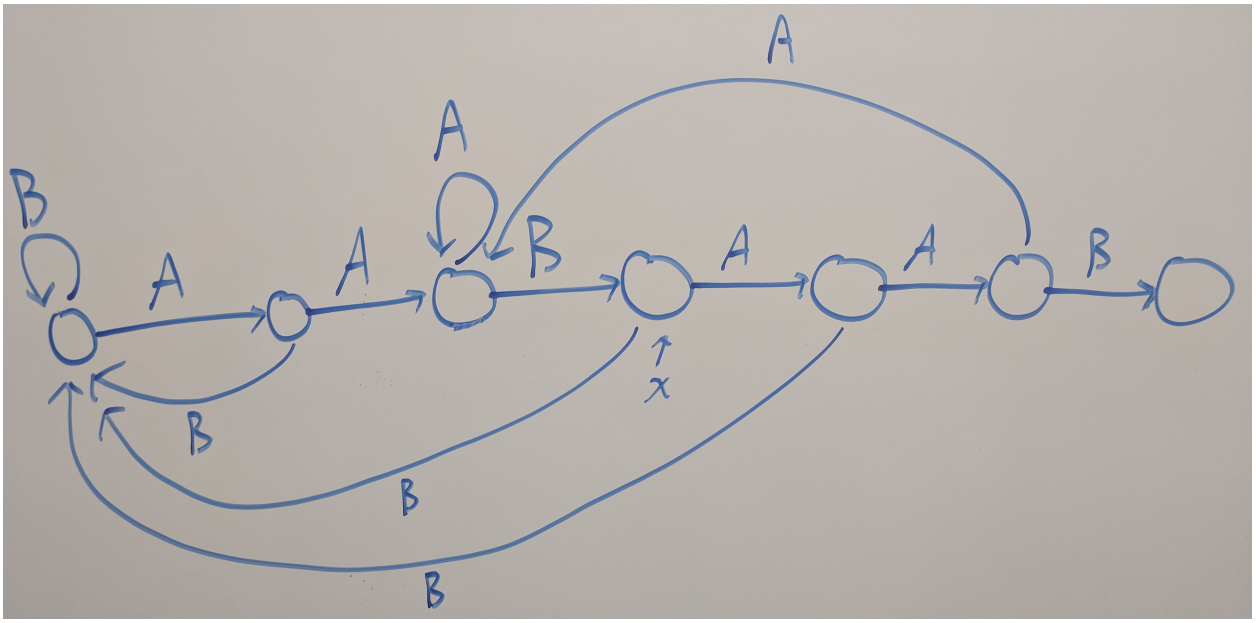
Flow network with given flow:



Answer:



7. [8 marks] Draw the DFA corresponding to the pattern AABAAB, where the alphabet is {A,B}.



(please ignore the x and its arrow)

8. [6 marks] Consider the version of the Boyer-Moore algorithm that only uses the mismatched character heuristic. Construct an example (consisting of a text and a pattern) under which this algorithm performs roughly the same number of comparisons as the brute force algorithm. Recall that the brute force algorithm makes roughly $m \cdot n$ comparisons, where m is the length of the pattern and n is the length of the text, and you are to assume that m is much smaller than n.

Solution: Let the text be a very long sequence of A's. For example:

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Let the pattern be BAAAA.

Initially, the B of the pattern will align with the first A of the text. The comparison will proceed from right to left, until the mismatch of the pattern's B with the text's initial A. The pattern then slides to the right by one character, and this process repeats over and over again. This behavior is essentially the same as the brute force algorithm in terms of runtime.

9. [10 marks] *In preparation for the actual final exam:* Think about a way to extend the Rabin-Karp algorithm to search a text string for an occurrence of either of 2 patterns *without* comparing the hash of each substring of the text to the hash of each of the two patterns. You may assume that both patterns have the same length, and it is only necessary to indicate whether or not there exists a pattern that occurs in the text (as opposed to specifying which pattern(s) occur).

10. [5 marks] Is the code shown below a prefix code? If not, indicate a single letter (among {c,h,e,k,i,t}) which, if removed, renders the code a prefix code.

    C(`c`) = 100
    C(`h`) = 01
    C(`e`) = 101
    C(`k`) = 0001
    C(`i`) = 010
    C(`t`) = 011

    Solution: No it is not a prefix code, since the codeword for `h` is a prefix of both the codeword for `i` and the codeword for `t`. Removing `h` will render the code a prefix code.

11. [5 marks] Recall that in the knapsack problem, we are given n items $\{1, \ldots, n\}$, and each item has a nonnegative weight w_i and a value v_i (for $i = 1, \ldots, n$). We also are given an upper bound W. The goal is to select a subset S of the n items so that $\sum_{i \in S} v_i$ is the maximum possible, subject to the constraint that $\sum_{i \in S} w_i \leq W$.

    For any $j \in \{0, 1, \ldots, n\}$ and $w \geq 0$, let $\text{OPT}(j, w)$ denote the value of the optimal solution among items $\{1, \ldots, j\}$ subject to the sum of the weights of the items used being at most w.

    What is the key recurrence which allows us to compute $\text{OPT}(j, w)$ via optimal solutions to subproblems?

$$\text{OPT}(j, w) = \begin{cases} 0 & \text{if } j = 0 \\ \text{OPT}(j-1, w) & \text{if } w_j > w \\ \max\{\text{OPT}(j-1, w), v_j + \text{OPT}(j-1, w - w_j)\} & \text{otherwise} \end{cases}$$

    For this question, nearly full marks would be awarded for only stating the the last line (the "otherwise" case).

12. [6 marks] Consider a nationwide wireless network band allocation problem for the country of New Zealand. There is a range of frequencies available, starting from $f_{\min}$ and going all the way up to $f_{\max}$. Each of n companies would like to request a contiguous interval that is a subset of this band. In particular, company j would like to request the interval $[\ell_j, u_j]$, containing all the frequencies in between the lower bound ℓ_j and the upper bound u_j. Since these intervals may overlap, not all companies can be allocated their request. The government of New Zealand, being a friendly bunch, wants to satisfy as many companies as possible by maximizing the number of companies that receive their requested intervals. Describe an algorithm design paradigm that may be used to solve this problem.

Solution: This problem can be solved using a greedy strategy. The problem is exactly analogous to the interval scheduling problem, for which the earliest-finish-time first algorithm is optimal. The finish time for the j^{th} job in the interval scheduling problem corresponds to the upper bound u_j for the j^{th} company.

For extra marks, design an algorithm for solving this problem.

Solution: This problem is equivalent to an interval scheduling problem. It can be solved by sorting the companies in the order of increasing u_j; let the sorted sequence be $u_{j_1}, u_{j_2}, \ldots, u_{j_n}$. We will incrementally augment a set A (initially empty) by one company index at a time until the set A is equal to an optimum solution. We say that a company j is compatible with a company A if, for all $k \in A$, the interval $[\ell_j, u_j]$ does not intersect the interval $[\ell_k, u_k]$.

The following code constructs the optimal set A, which is initialized to the empty set.

For $i = 1$ to n

    If job j_i is compatible with A

        $A = A \cup \{j_i\}$

(Note: an alternative solution is to use the above loop but in the case where the companies are sorted in order of *decreasing* ℓ_j.)