Massachusetts Institute of Technology
6.046J/18.410J: Introduction to Algorithms
Professors Piotr Indyk and Bruce Tidor

Practice for Final Exam (Solutions)

- Do not open this quiz booklet until you are directed to do so.
- This final ends at 12:00 P.M.
- This exam is closed book. You may use three sides of handwritten $8\frac{1}{2}'' \times 11''$ crib sheets.
- When the quiz begins, write your name on the top of ***EVERY*** page in this quiz booklet, because the pages will be separated for grading.
- Write your solutions in the space provided. If you need more space, write on the *back* of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem.
- Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- When describing an algorithm, describe the main idea in English. Use pseudocode only to the extent that it helps clarify the main ideas.
- Good luck!

Name: _____

Please circle your TA's name and recitation:

Nitin **Brian** **Mihai** **Yoav**

10am **11am** **12pm** **1pm** **2pm**

Problem 1. Short questions

In the following questions, fill out the blank boxes. In case more than one answer is correct, you should provide the **best known** correct answer. E.g., for a question "Sorting of n elements in the comparison-based model can be done in $\boxed{}$ time", the best correct answer is $O(n \log n)$. No partial credit will be given for correct but suboptimal answers. However, you **do not** have to provide any justification.

(a) Consider a modification to QUICKSORT, such that each time PARTITION is called, the median of the partitioned array is found (using the SELECT algorithm) and used as a pivot.

The worst-case running time of this algorithm is: $\boxed{}$

Answer: $\Theta(n \log n)$. `Reason: the median can be found in` $O(n)$ `time, so we have the running time recurrence` $T(n) = 2T(n/2) + O(n)$.

(b) If a data structure supports an operation `foo` such that a sequence of n `foo`'s takes $\Theta(n \log n)$ time to perform in the worst case, then the amortized time of a `foo` operation is $\Theta\left(\boxed{}\right)$, while the actual time of a single `foo` operation could be as high as $\Theta\left(\boxed{}\right)$.

Answer: `Amortized time` $\Theta(\log n)$, `worst-case time` $\Theta(n \log n)$.

(c) Does there exist a polynomial time algorithm that finds the value of an $s-t$ minimum cut in a directed graph? $\boxed{}$ (Yes or No)

Answer: `Yes. Reason: The value of an` $s - t$ `minimum cut is equal to the value of the` $s - t$ `maximum flow, and there are many algorithms for computing` $s-t$ `maximum flows in polynomial time (e.g. the Edmonds-Karp algorithm).`

Problem 2. True or False?

For each of the questions below, circle either **T** (for True) or **F** (for False). **No explanations are needed.** Incorrect answers or unanswered questions are worth zero points.

T **F** By the master theorem, the solution to the recurrence $T(n) = 3T(n/3) + \log n$ is $T(n) = \Theta(n \log n)$.

Answer: `False`

T **F** Computing the median of n elements takes $\Omega(n \log n)$ time for any algorithm working in the comparison-based model.

Answer: `False`

T **F** Every binary search tree on n nodes has height $O(\log n)$.

Answer: `False`

T **F** Given a graph $G = (V, E)$ with cost on edges and a set $S \subseteq V$, let (u, v) be an edge such that (u, v) is the minimum cost edge between any vertex in S and any vertex in $V - S$. Then, the minimum spanning tree of G must include the edge (u, v). (You may assume the costs on all edges are distinct, if needed.)

Answer: `True`

T **F** Let T be a minimum spanning tree of G. Then, for any pair of vertices s and t, the shortest path from s to t in G is the path from s to t in T.

Answer: `False`

T **F** If a problem L_1 is polynomial time reducible to a problem L_2 and L_2 has a polynomial time algorithm, then L_1 has a polynomial time algorithm.

Answer: `True`

Problem 3. True or False and Justify.

For each of the questions below, circle either **T** (for True) or **F** (for False). Then give a brief **justification** for your answer. Your answers will be evaluated based on the justification, and **not** the **T**/**F** marking alone.

T **F** There exists a data structure to maintain a dynamic set with operations Insert(x,S), Delete(x,S), and Member?(x,S) that has an expected running time of $O(1)$ per operation.

Answer: `True. Use a hash table.`

T **F** The total amortized cost of a sequence of n operations (i.e., the sum over all operations, of the amortized cost per operation) gives a lower bound on the total actual cost of the sequence.

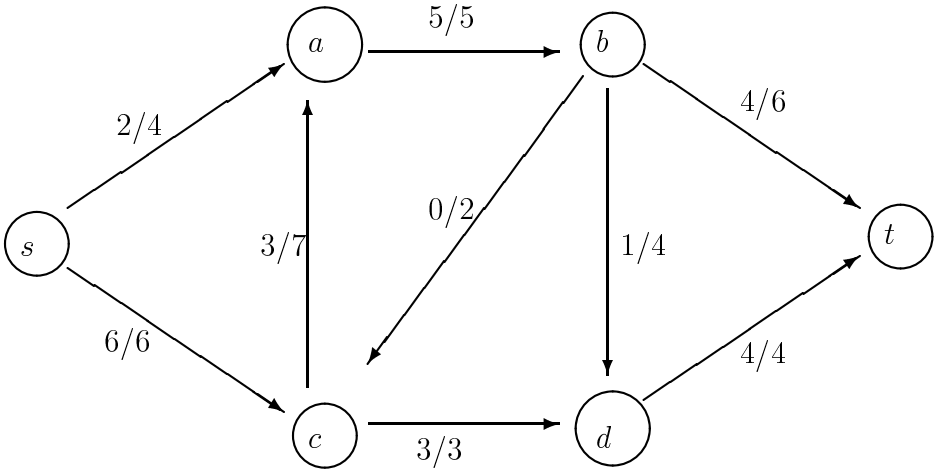
Answer: `False. This only gives an upper bound on the actual cost.`

T F Let $G = (V, E)$ be a weighted graph and let M be a minimum spanning tree of G. The path in M between any pair of vertices v_1 and v_2 must be a shortest path in G.

Answer: `False. Consider the graph in which` $V = \{a, b, c\}$ `and the edges are` $(a, b), (b, c), (c, a)$`. The weight of the edges are 3,3 and 4 respectively. The MST is clearly` $(a, b), (b, c)$`. However the shortest path between` a `and` c `is of cost 4 not 6 as seen from the MST.`

T F The figure below describes a flow assignment in a flow network. The notation a/b describes a units of flow in an edge of capacity b.

True or False: The following flow is a maximal flow.



Answer: `True. The flow pushes 8 units and the cut` $\{s, a, c\}$ `vs.` $\{b, d, t\}$ `has capacity 8. The cut must be maximum by the Max-Flow Min-cut theorem.`

Problem 4. Typesetting

Suppose we would like to neatly typeset a text. The input is a sequence of n words of lengths $l_1, l_2, \cdots, l_n$ (measured in the number of fixed-size characters they take). Each line can hold at most P characters, the text is left-aligned, and words cannot be split between lines. If a line contains words from i to j (inclusive) then the number of spaces at the end of the line is $s = P - \sum_{k=i}^{j} l_k - (j - i)$. We would like to typeset the text so as to avoid large white spaces at the end of lines; formally, we would like to minimize the sum over all lines of the square of the number of white spaces at the end of the line. Give an efficient algorithm for this problem. What is its running time?

Answer:

We solve the problem with dynamic programming. Let $A[j]$ denote the optimal ''cost'' (that is, the sum of the square of number of trailing white space characters over all lines) one may achieve by typesetting only the words $1 \ldots j$ (ignoring the remaining words). We can express $A[j]$ recursively as follows:

$$A[j] = \min_{i<j : T[j]-T[i] \leq P} A[i] + (P - (T[j] - T[i]))^2,$$

where $T[j] = \sum_{i=1}^{j} l_i$. A table of the values $T[1 \ldots n]$ can be initially computed in $O(n)$ time. The equation above says the following: in order to optimally typeset words $1 \ldots j$, we must first optimally typeset words $1 \ldots i$ for some $i < j$, and then place the remaining words $i + 1 \ldots j$ on the final line.

Using dynamic programming, we compute each value $A[j]$ in sequence for all $j = 1 \ldots n$. Each value is requires $O(n)$ time to compute, for a total running time of $O(n^2)$. After termination, $A[n]$ will contain the value of the optimal solution; we can reconstruct the solution itself (that is, the locations where we should insert line breaks) by maintaining ''backpointers'' as is usually done with dynamic programming.

Problem 5. Princess of Algorithmia (2 parts)

Political upheaval in Algorithmia has forced Princess Michelle X. Goewomans to vacate her royal palace; she plans to relocate to a farm in Nebraska. The princess wants to move all of her possessions from the palace in Algorithmia to the Nebraskan farm using as few trips as possible in her Ferrari. For simplicity, let us assume that the princess's Ferrari has size 1, and that her n possessions $x_1, x_2, \ldots, x_n$ have real number sizes between 0 and 1. The problem is to divide the princess's possessions into as few carloads as possible, so that all her possessions will be transported from Algorithmia to Nebraska and so that her Ferrari will never be overpacked. It turns out this problem is NP-hard.

Consider the following **first-fit** approximation algorithm. Put item x_1 in the first carload. Then, for $i = 2, 3, \ldots, n$, put x_i in the first carload that has room for it (or start a new carload if necessary). For example, if $x_1 = 0.2$, $x_2 = 0.4$, $x_3 = 0.6$, and $x_4 = 0.3$, the first-fit algorithm would place x_1 and x_2 in the first carload, x_3 in the second carload, and then x_4 in the first carload (where there is still enough room). Note that all decisions are made offline; we divide the princess's n possessions into carloads before any trips have actually been made.

(a) The princess's charming husband, Craig, has asserted, "The first-fit algorithm will always minimize the total number of car trips needed." Give a counterexample to Craig's claim.

Answer:

Let $n = 4$ and let $x_1 = 0.3$, $x_2 = 0.8$, $x_3 = 0.2$, and $x_4 = 0.7$. The optimal number of car trips needed is 2, while the first-fit algorithm produces 3 car trips.

(b) Prove that the first-fit algorithm has a ratio bound of 2. (*Hint:* How many carloads can be less than half full?)

Answer:

Consider a carload u of size p which is less than half full, i.e., $p < 0.5$. Let x_i be the first item to go into the next carload v. Then, due to the nature of the first-fit algorithm, it follows that $p+x_i > 1$ and $x_i > 0.5$. Hence, carload v is more than half full. For calculation purposes, we can transfer a portion of the load of item x_i from carload v to carload u to make carload u exactly half full while keeping carload v at least half full. This is because $0.5-p < x_i-0.5$. After we perform this operation, every carload is at least half full. Thus, if h is the number of carloads produced by the first-fit algorithm, then $\sum_{i=1}^{n} x_i \geq 0.5h$. Note that the optimal number of carloads OPT is at least $\sum_{i=1}^{n} x_i$. Hence, $h/OPT \leq 2$, which proves that the first-fit algorithm has a ratio bound of 2.

Problem 6. Division of Power in King Arthur's Court (2 parts)

A large number of problems that we now refer to as optimization problems, actually date back to King Arthur's time. King Arthur's court had many knights. They were the source of many of his problems, some computational and others not. King Arthur often sought the advice of his powerful wizard — Merlin — to solve these computational problems. It is rumored that the mighty Merlin underwent a voyage in a time machine, took 6.046 in the Fall of 2000 and used these skills to solve many of the challenges posed to him.

In the two parts of this question, we'll see some of the problems posed to him.

(a) Ruling the land:

King Arthur's court had n knights. He ruled over m counties. Each knight i had a quota q_i of the number of counties he could oversee. Each county j, in turn, produced a set S_j of the knights that it would be willing to be overseen by. The King sets upon Merlin the task of computing an assignment of counties to the knights so that no knight would exceed his quota, while every county j is overseen by a knight from its set S_j.

(i) Show how Merlin can employ the Max-Flow algorithm to compute the assignments. Describe the running time of your algorithm. (You may express your running time using function $F(v, e)$, where $F(v, e)$ denotes the running time of the Max-Flow algorithm on a network with v vertices and e edges.)

Answer: We make a graph with $n+m+2$ vertices, n vertices $k_1, \ldots, k_n$ corresponding to the knights, m vertices $c_1, \ldots, c_m$ corresponding to the counties, and two special vertices s and t.
We put an edge from s to k_i with capacity q_i. We put an edge fro k_i to c_j with capacity 1 if county j is willing to be ruled by knight i.
We put an edge of capacity 1 from c_j to t.
We now find a maximum flow in this graph. If the flow has value m, then there is a way to assign knights to all counties, as argued next.
Since this flow is integral, it will pick one incoming edge for each county c_j to have flow of 1. If this edge comes from knight k_i, then county j is ruled by knight i.
The running time of this algorithm is $F(n + m + 2, \sum_j |S_j|)$.

(ii) Merlin runs his algorithm (from Part (i)) but the algorithm does not produce an assignment that fits the requirements. King Arthur demands an explanation. Merlin explains his algorithm to King Arthur, and King Arthur is convinced that the algorithm is correct. But he does not believe Merlin has executed this algorithm correctly on the given instance of the problem. He wants Merlin to convince him that no assignment is possible in this particular case. Based on your understanding on Max-Flow, what proof would you suggest? (Note that your proof strategy should work for every instance of the problem for which a satisfactory assignment does not exist.)

Answer: If there is no flow of size m, then there must a cut in this graph of capacity less than m. Looking at the structure of the cut, we note that this gives a set T of counties such that if the U is the union of the sets S_j for $j \in T$, then the sum $\sum_{i \in U} q_i$ is less than $|T|$. (I.e., there is a subset of counties such that the quotas of the knights that they are willing to be ruled by is smaller than the number of counties in the subset.) Merlin can present the sets T and U to Arthur explaining why T can not be assigned knights.

(b) Conflict resolution: At an annual meeting of the knights, King Arthur notes that several of knights have started to develop conflicts. Sending his undercover agents, he has managed to find out a list of all possible duelling pairs. He now wishes to exile a small set of knights away so that no duelling pair remains at the annual meeting. Merlin is now set with this task of finding out which knights to send away. Being a perfectionist, Merlin wishes to find the smallest set he could send into exile. However, after much thought, he is unable to find the optimal solution using any efficient algorithm.

(i) Explain why? (I.e., what problem is Merlin trying to solve and why is he unable to do it.)

Answer: `The set of knights to be exiled form a vertex cover in the incompatibility graph. Finding the smallest such cover is NP-complete and this is why Merlin is unable to solve the problem efficiently.`

(ii) How could Merlin weaken his goals to find a reasonable solution?

Answer: `He can use a 2-approximation algorithm for Vertex Cover and thus exile a subset of knights of size at most twice the optimum number.`