CSE 551 Midterm Test 1 Solutions
28 September 2021, 1:30-2:45 p.m.

Last Name	SAMPLE	ASU ID	3336669990
First Name(s)	Ima	Exam #	1
I attest that I am the student whose name and number is listed on this paper. I have read the instructions given on this page. Signature:			

Regrading of Midterms

If you believe that your grade has not been recorded correctly, return the entire paper to the instructor with a short note indicating what you believe to be the error. Other than for that reason, test grades are almost never changed. If you believe that you did not receive the proper credit, first **read these sample solutions carefully** to see if you can understand the answer to your concern. If that does not resolve it, write a clear explanation of why you believe the grade is in error and submit that, along with the entire test paper, to the instructor. Please do not discuss in your explanation how your solution is like that of another student, as FERPA legislation makes it impossible for me to discuss one student's work with another. Please take into account that many papers were graded, and it is impossible to change the grade on one paper without giving every other student the same opportunity. If you nevertheless want the paper regraded, be advised that the entire paper will be regraded and the grade may go up, stay the same, or go down. The new grade will be final. It is a violation of the Academic Integrity Policy to request a grade change simply because you need or want a higher grade.

If you require a clarification of the sample solutions (not a grade change or review as discussed above), ask in the office hours of the TA or instructor. You will be asked whether you have read the sample solution and to indicate what precisely is unclear to you about it, so **read these sample solutions carefully** first. Under no circumstances can anyone change a grade other than the instructor, so do not ask a TA to do so.

Grade review requests, whether submitted as described above or not, will not be considered if received after the start of class on 14 October 2021. If you cannot get it to me in class, you can ask the staff in the SCAI Main Office to put it in my mailbox. I need the original, not a scan or photo.

Instructions

Do not open the exam until you are instructed to do so. You have 75 minutes to complete the exam. When the time is over, stop writing. No books, notes, electronic devices, or other aids are permitted. All devices must be OFF; put them in your backpack or with your books. *This includes phones, watches, etc.* Place backpacks, notes, texts, etc. away from your work space, e.g., at the side or front walls. You only need your pens. Write all answers on the examination paper itself. Give one answer to each question; multiple answers to a single question will result in lower grades. Your answer should be both precise and concise; do not include unnecessary information. BUDGET YOUR TIME WELL! SHOW ALL WORK!

Sample, Ima					
Question	Mark	Out Of	Question	Mark	Out Of
1		5	2		5
3		5	4		5
5		10	6		10
7		10	8		10
Total		60			

Question 1. [5 marks, $\frac{1}{2}$ for each response.] *Fill in each box with a nonnegative integer.* Twelve workers have been assigned to twelve jobs using the Gale-Shapley algorithm. Each worker has provided a complete ranking of the jobs and each job has provided a ranking of the workers. Workers propose. Fill in each of the blanks that follow with an integer. The number of pairs in a stable matching is $\boxed{12}$. The number of unstable pairs in an arbitrary matching is at most $\boxed{132}$. The number of workers who are not assigned to their most preferred valid job can be any number between $\boxed{0}$ and $\boxed{0}$. The number of workers who are not assigned to their most preferred job (whether valid or not) can be any number between $\boxed{0}$ and $\boxed{12}$. The number of jobs that are not assigned their most preferred valid worker can be any number between $\boxed{0}$ and $\boxed{12}$. The number of proposals made must be a number that is at least $\boxed{12}$ and at most $\boxed{133}$.

Question 2. [5 marks, graded as max(5−#incorrect,0)] *Fill in each box with a capital letter, chosen as follows.* Some upper bounds in $O()$ notation on functions of n are A: $O(1)$; B: $O(\log n)$; C: $O(n)$; D: $O(n \log n)$; E: $O(n^{\log_3 2})$; F: $O(n^{\log_2 3})$; G: $O(n^2)$; H: $O(2^n)$. For each of the following recurrences (with c a constant), choose the tightest of these bounds for each recurrence given, and indicate the corresponding letter in the box provided.

$$T(n) = T(n/2) + c \qquad \boxed{B}$$
$$T(n) = 2T(n/3) + cn \qquad \boxed{C}$$
$$T(n) = 3T(n/2) + cn \qquad \boxed{F}$$
$$T(n) = nT(n/2) + cn \qquad \boxed{H}$$
$$T(n) = 4T(n/3) + cn \qquad \boxed{F}$$
$$T(n) = T(n/2) + T(n/4) + T(n/8) + cn \qquad \boxed{C}$$
$$T(n) = 3T(n/3) + cn \qquad \boxed{D}$$

Question 3. [5 marks] We have been given an undirected graph $G = (V, E)$ in which every edge e has a positive integer weight $w(e)$; all weights are distinct. My friend found shortest paths for G from a source node s. But I just realized that I left out one of the edges, f with positive weight $w(f)$, when I provided the graph to my friend. Now I am worried that the shortest paths may not be right for the graph with f added. Which of the following might happen? *Mark each box provided with* $\surd$ *if the answer is correct. If incorrect, leave it blank or mark it with an* $\times$.

$\boxed{\surd}$ The shortest path distance changes to exactly one node.

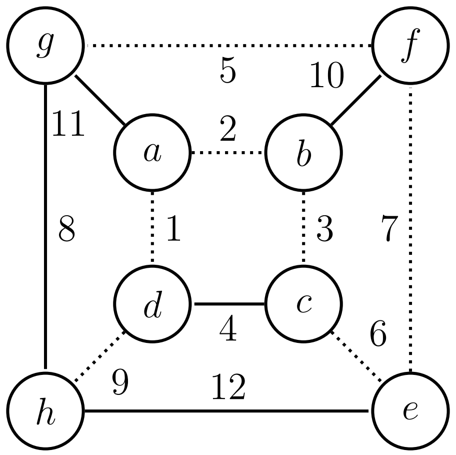
[✓] The shortest path distance changes to exactly $|V| - 1$ nodes.

[ ] The shortest path distance changes to all nodes.

[✓] The shortest path distance does not change to any node.

[ ] The shortest path distances can no longer be computed.

Question 4. [5 marks] Consider the following graph with positive edge weights as shown.



(a) Using **only** the cycle property, show that the dotted edges do not form a minimum weight spanning tree. In the cycle (d, h, g, f, e, c) the edge $\{d, h\}$ has the largest weight, so it cannot be in an MST. Other cycles are possible instead.

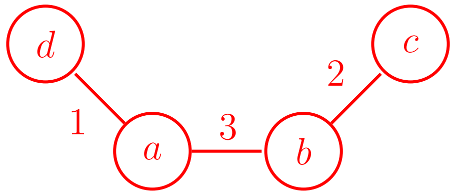
(b) Using **only** the cut property, show that the dotted edges do not form a minimum weight spanning tree. In the cutset with edges $\{\{d.h\}, \{e, h\}, \{g, h\}\}$, edge $\{g, h\}$ has smallest weight, so it must be in the MST.

Question 5. [10 marks] Let $G = (V, E)$ be a graph with integer edge weights. A *subtree* of G is a graph $T = (W, F)$ with $W \subseteq V$ and $F \subseteq \{\{x, y\} : \{x, y\} \in E \text{ and } x, y \in W\}$ so that T is connected and contains no cycles. Given an edge-weighted graph G and an integer k as input, we want to find a subtree of G having exactly k vertices whose total edge weight is as small as possible, or report that no such subtree exists. Describe the most efficient algorithm that you can, and briefly analyze its running time in $O()$ notation supposing that G has n vertices and m edges.

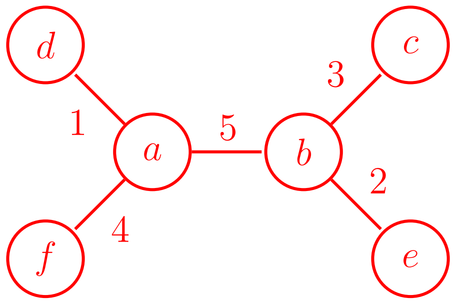
The problem statement does not require G to be connected. Apply breadth-first search to find the components of G and determine their sizes. Delete all components having fewer than k vertices. If no components remain, report that there is no subtree on k vertices and stop. Otherwise, apply a method for connected graphs to each of the remaining components and report the minimum. This all takes $O(n+m)$ time and reduces the problem to the same problem for *connected* graphs.

The problem statement does not require that all edge weights be distinct. We can perturb each edge weight by a small random amount without changing the solution, so we assume that all edge weights are distinct.

You might try Kruskal, making sure it never makes a component on more than k vertices. You might try reverse delete, stopping when there is no component on more than k vertices. But what do they do on the following graph G_1 when $k = 3$?



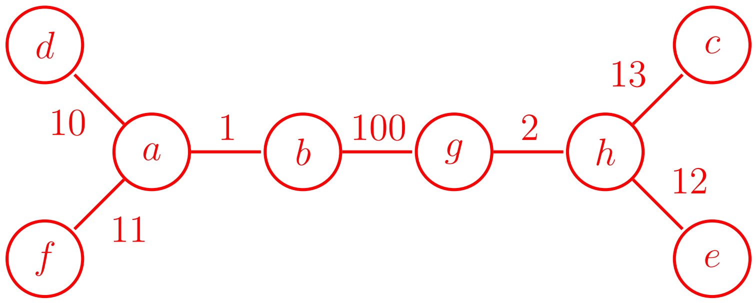
So what about Prim, stopping when the tree built has k vertices? It does not work on G_1 if you start with root b or c, but does work with root a or d. But what about another graph G_2:



Starting at any root with $k = 4$ you get weight 10 in total. But you should get total weight 8. Sigh....

I am running out of time, so what can I do? For each subset S of the $\binom{n}{k}$ subsets of V on k vertices, form the subgraph G_S containing all edges with both ends in S. If G_S is connected, compute its MST. Keep the minimum weight of all of the MSTs found. This is an exponential time algorithm, but at least it works!

Some students suggested first finding an MST and then repeatedly deleting leaves for which the edge weight to the leaf is maximum, until k vertices remain. What about this graph G_3?



With $k = 4$ the method gets weight 103, but it should get 22.

Question 6. [10 marks] The Gale-Shapley algorithm has been used to find a stable matching between n men and n women (each giving a complete preference list for the other). Men propose. After running the algorithm, one man changes his mind about his preferences. For all $n \geq 2$, give an example in which the maximum number of pairs change in the stable matching as a result. Say what this maximum number is, and justify why it is the maximum.

This is not asking about the number of proposals, the number of unstable pairs, or the number of valid partners. Also, it is not saying that Gale-Shapley is restarted or rerun after the change.

The n men are $M_1, \ldots, M_n$, the women are $W_1, \ldots, W_n$. I give part (but enough) of the initial preference lists. For $1 \leq i \leq n-1$, M_i has first choice W_i and second choice W_{i+1} and man M_n has first choice W_1. For $2 \leq i \leq n$, W_i has first choice M_{i-1} and second choice M_i, and W_1 has first choice M_n and second choice M_1. So what happens? In the GS stable matching, M_i is matched to W_i for $1 \leq i \leq n-1$ in the first $n-1$ proposals. Then M_n proposes to W_1 and she trades up. Now for $1 \leq i \leq n-2$, M_i proposes to W_{i+1} and she trades up. Finally M_{n-1} proposes to W_n and she accepts and the stable matching is found.

Now M_n just changes his preference to W_n instead of W_1. All n pairs of the stable matching change. This is the maximum because there are only n pairs.

Some students told me the minimum, not the maximum. Some students said that the maximum is 2 and some said $n-1$. MANY students gave an example for $n = 2$ or $n = 3$ and some also said that this could be generalized or was easy to generalize for all n. Examples are useful ONLY to illustrate a general solution after you have given it; they do NOT replace the general solution needed. Most of the time, there are infinitely many ways to generalize one small example, and the example does not tell you which one is intended (if any). When a question asks for a general solution, give a general solution. If the generalization from an example is easy, just do it.

Question 7. [10 marks] An election has just been held. There have been n votes cast (in no particular order), each voting for one of m candidates. Our task is to determine whether any candidate has won more than 50% of the votes, and if so, which candidate. Devise the most efficient algorithm that you can, stating its running time in $O()$ notation, and establishing its correctness. *You can use any algorithm presented in lecture as part of your solution, but your runtime analysis must include the time for such a method.*

I give a linear time algorithm, as follows:

1. Compute the median d of the list of n votes (ordering the candidate names in any fixed ordering)
2. Compare the median d to each of the n votes, computing the number c that are equal to d.
3. If $c > \frac{n}{2}$ answer YES, CANDIDATE d has a majority; else NO MAJORITY.

Runtime: Use the $O(n)$ median-of-median algorithm to do step (1). Step (2) runs through the n votes once, comparing each to d, so it takes $O(n)$ time. Step (3) takes constant time, so we have an $O(n)$ time algorithm.
Correctness: We only report a majority when d agrees with more than $\frac{n}{2}$ of the votes, so if the algorithm says YES, there is a majority winner. On the other hand, no element that is not the median can get a majority because if it is in the first half but is not equal to the median, more than half the votes are for later candidates. If it is in the last half but is not equal to the median, the situation is symmetric.

There is no limitation in the question that $m < n$ — there may be many more candidates than votes. So if you make an array indexed by the candidates, how can you initialize it in fewer than m steps? If you get an $O(m+n)$ algorithm this is much worse than $O(n)$ when $m \gg n$.

If, instead of a linear array, you use a hashing scheme (a "hashmap"), can you tally the votes for each candidate in $O(n)$ time in the worst case, without making extra assumptions about the candidates? How?

Some students asserted that they could use a radix or bucket sort to sort the votes in $O(n)$ time. Without unwarranted assumptions about the candidates, I see no way to do this.

Many students sorted using a conventional sort, or adapted the divide-and-conquer in merge-sort, to get an $O(n \log n)$ time method. This works, but as we have seen, there is a faster method.

Two students knew of an another algorithm, the *Boyer-Moore Algorithm*. Here it is:

1. Initialize an element d arbitrarily, and a counter c with $c = 0$ initially
2. For each element x of the input
   (a) If $c = 0$, set $d = x$ and $c = 1$
   (b) else if $d = x$, set $c = c + 1$
   (c) else set $c = c - 1$
3. Read the input again to get the count γ of the elements equal to d, and the total number n of elements.
4. If $2\gamma > n$, output "Majority element is d" else output "No Majority"

Making two passes over the input gives an $O(n)$ time algorithm. I did not expect anyone to know this method, but it is fine if you did.

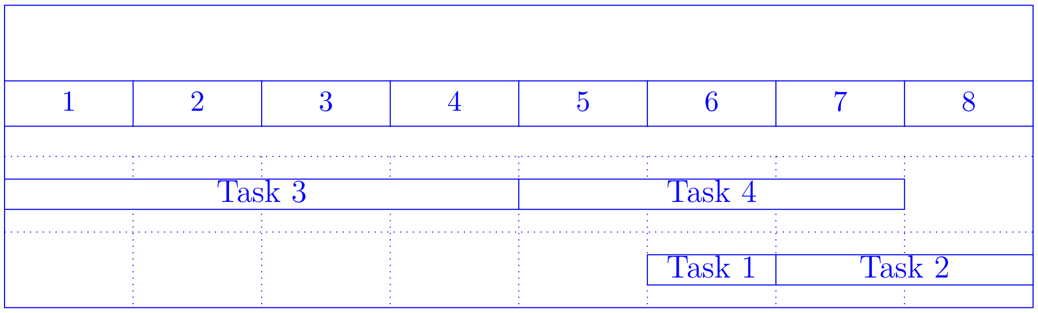
Question 8. [10 marks] Interval partitioning gives the minimum number of cores required to run all jobs in the queue. In practice, we may only have k cores, but more than k cores are required to run every task. In order to run as many jobs as possible, the following scheme is proposed.

> Use the interval partition algorithm described in class to compute the fewest cores required to complete all jobs. Sort the cores from the largest number of jobs completed to the smallest number of jobs completed. Run the jobs on the top k cores.

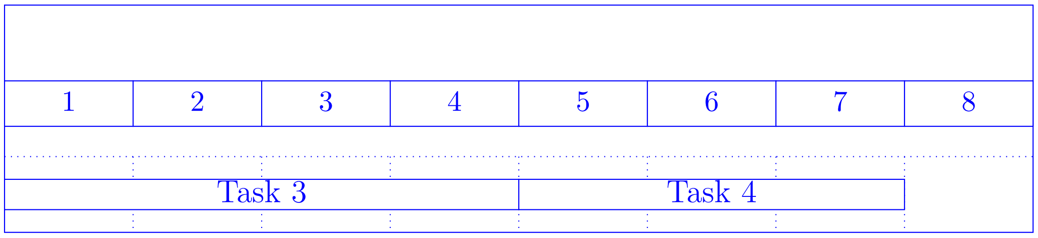
Prove or disprove: This method determines the largest number of jobs that can be run on k cores.

This was on a homework. The counterexample used here is somewhat simpler than the one in the homework solution.

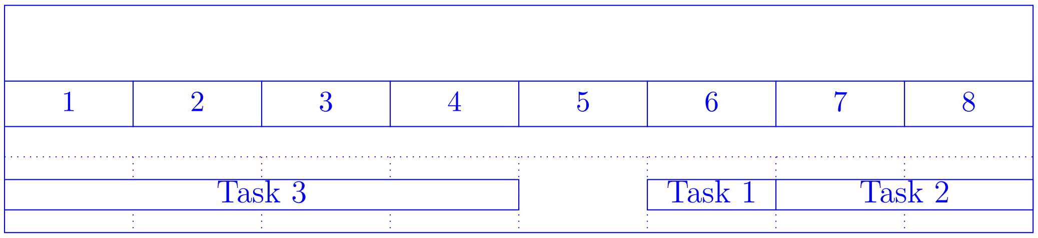
We *disprove*. We give a counterexample for $k = 1$. It has four jobs for which interval partitioning yields:



Select the top core to get two jobs completed:



But we can get three!


1
2
3
4
5
6
7
8
Task 3
Task 1
Task 2