CSCI 570

Exam 1 Rubrics

Problem 1: True or False (20 pts)

Q1-10 (Random Questions)

Rubric: 2 points each correct Answer. Max 20 points

[True/False]
There are at most two stable matching solutions in a stable matching problem.

[True/False]
If f(n) = O(g(n)), then for large values of n beyond some constant n_0, we always have f(n)<=g(n).

[True/False]
In a graph with positive edge weights, if we add the same positive weight on every edge, Dijkstra's algorithm will still find the nodes in the same order, when starting from the same source node.

[True/False]
Dijkstra's algorithm is able to find the Shortest Path in directed and undirected graphs with positive edge weights.

[True/False]
The recurrence $T(n) = 16T(n/4) - n^2 \log n$ can be solved using the Master Theorem.

[True/False]
There is a path from any point to any other point in a connected undirected graph.

[True/False]
The merge operation in a binomial heap takes $\theta(\log n)$.

[True/False]
In an undirected connected graph, if the cost of all edges is the same, we can use DFS to find a minimum spanning tree.

[True/False]
A correctly constructed dynamic programming algorithm may have an exponential running time.

[True/False]
In the Bellman-Ford algorithm, if every node keeps a pointer to the neighbor that gives it a shortest distance to the destination, a top down pass will not be required to find shortest paths from all nodes to t.

Problem 2: Multiple Choice (25 pts)

Q11-15 (Random Questions)

Rubric: 5 points each correct Answer. Max 25 points. Please note that the choices might also be in random order, so compare your answer directly to the case description.

1. Which of the following has the highest time complexity?

   A. θ(n^3)

   B. θ(n^2.5*(log n)^4)

   C. θ(1.01^n)

   D. θ(3^(log n))

2. Which of the following algorithms has a worst case run time complexity that is O($n^{1.5}$)

   A. Gale-Shapley—n representing the number of men and women

   B. Merge sort – n representing the size of the array

   C. Prim algorithm on a dense graph – n representing the number of nodes

   D. None of the above

3. Assume we have the following operations available on a data structure (Put, Get, Reorder). We know that Put takes constant time in the worst case and there is a worst-case sequence of *n* Put, Get and Reorder operations that takes exactly *5n log n* time. Which of the following statements is correct? Select all correct statements

   A. Amortized cost of Get is O(log n)
   B. Amortized cost of Put is O(log n)
   C. Amortized cost of Reorder is O(log n)
   D. None of the above

4. Consider an instance of a stable matching problem where there are three men, M1, M2, M3 and three women, W1, W2, W3. Their preference lists are given below – in descending order of preference:

   M1: W1, W2, W3

   M2: W1, W3, W2

   M3: W2, W3, W1

W1: M2, M1, M3

W2: M3, M1, M2

W3: M1, M2, M3

Which one of the following is a stable matching?

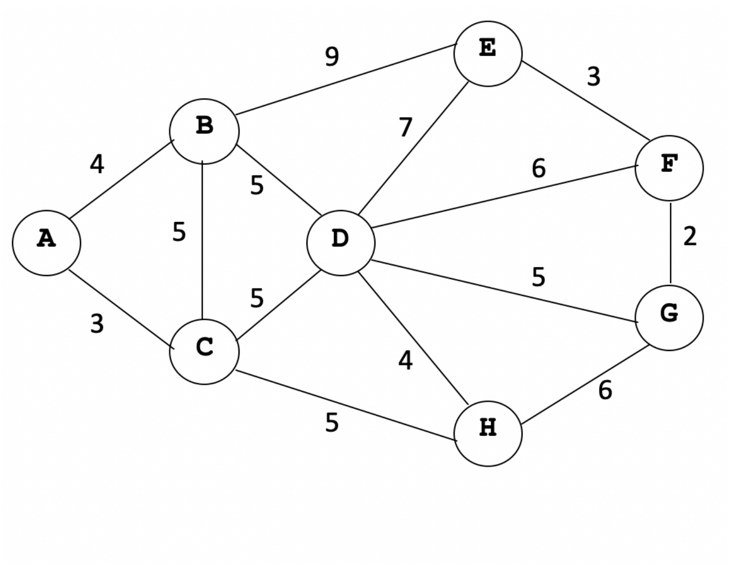
A. (M1, W3), (M2, W2), (M3, W1)

B. (M1, W3), (M2, W1), (M3, W2)

C. (M1, W1), (M2, W2), (M3, W3)

D. (M1, W1), (M2, W3), (M3, W2)

5. The total edge cost for the minimum spanning tree shown below is:



A. 26

B. 27

C. 28

D. 29

Problem 3 (10 pts)

Q16-17

Let be an $m \times n$ matrix of integers. Given an integer $k \leq m \times n$, show that the k biggest elements of the matrix can be found in $O(k \log(m \times n) + m \times n$.

a) Describe your algorithm (5 pts)

Describing : Construct Max_heap on all the elements 3 pts
Describing: k extract_max actions 2 pts
Inconsistency in the details: up to -4 pts

b) Analyze the complexity of your algorithm (5 pts)

Describing time complexity for Construct Max_heap 3 pts
Describing time complexity for k extract_max actions 2 pts
Inconsistency in the details: up to -4 pts

Problem 4 (15 pts)

Q18

A circle is centered at the origin of 2-dimensional space (x and y axis), and points are selected randomly on the perimeter of the circle. Selected points are stored as an array V[1…n], where each element of the array is a point represented by its coordinates. You may assume that points have distinct coordinates in both dimensions. Let V[1] represent the point with the minimum coordinate and that the vertices V[1…n] are ordered counter clockwise on the perimeter of the circle.
Propose an efficient algorithm with the complexity of *O(log n)* to find the vertex with the maximum x coordinate. Describe your algorithm using pseudocode and show your complexity analysis. You do not need to provide any proof of correctness for your algorithm.

Solution.
Non-recursive version

```
1 a, b ← 1, n
2 while a < b do
3   mid ← ⌊(a + b)/2⌋
4   if V[mid] < V[mid + 1] then a ← mid + 1
5   if V[mid] > V[mid + 1] then b ← mid
6 return V[a]
```

Recursive version

```
1 function f(a,b)
2   if a < b
3     mid ← ⌊(a + b)/2⌋
4     if V[mid] < V[mid + 1] then return f(mid + 1, b)
5     if V[mid] > V[mid + 1] then return f(a, mid)
6   else return V[a]
7 Main function
8   return f(1, n)
```

$T(n) = T(n/2) + O(1)$, according to the master theorem, $T(n) = O(\log n)$.

Rubrics

Grading code	Points	Solution part
A	1	Line 1 in non-recursion and Line 8 in recursion pseudocode
B	2	Line 3 in pseudocode
C	3	Line 4 in pseudocode
D	3	Line 5 in pseudocode
E	2	Line 2 & 6 in pseudocode
F	2	$T(n) = T(n/2) + O(1)$
G	1	$T(n) = O(\log n)$

There are no partial credits in a single Grading Code A ~ G.

Pseudocode is mandatory to qualify for Grading Code A ~ E. Writing in plain English or other natural language is not eligible to claim A ~ E.

To qualify for Grading Code F & G, you must gain at least 60 percent (7.2 points) from A~E. There are no points for time complexity if your algorithm is wrong and you are analyzing the time complexity for a wrong algorithm.

When filing regrade requests, students must list *ALL* Grading Code A ~ G that are qualified and sum up the projected total grade. Otherwise, the regrade request will be rejected and there is no second chance to resubmit. All grading codes students claimed will be reviewed again.

Problem 5 (15 pts)

Q19-21

Mrs. T attends a mathematics competition with a team of n students. In the contest, each team is given a set of n questions, where the i th question has a difficulty of d(i). Each student j can handle a certain level of question difficulty s(j), i.e. student j can only solve the i th question iff d(i) <= s(j). Mrs T can assign at most one question to each student to solve.

a) Propose an algorithm to help Mrs. T to maximize her team score. In other words, assign questions to team students in the team such that she maximizes the number of problems that can be solved by her team. (6 pts)

b) What is the worst case run time complexity of your solution? (3 pts)
   A. θ(log n),
   B. θ(n),
   C. θ(n log n),
   D. θ(n^2),
   E. θ(n^2 log n),
   F. θ(n^3)

c) Prove the correctness of your algorithm. (6 pts)

Solution. Greedy algorithm. Sort questions by their difficulty level in an increasing order. Sort the students by their difficulty threshold in an increasing order. Assign the question with the lowest difficulty to the student with the lowest threshold if he/she can solve it.

Rubrics.

a) If sorting is done correctly but the student has forgotten to mention that the lowest difficult must be assigned to the student with the lowest threshold if he/she can solve (greedy) and the assignment is just direct matching then -1

c) If the idea mentions the idea but unable to complete the proof -2

Problem 6 (15 pts)

Q22-24

Justin and Randall work at a restaurant near USC. Customers who come to the restaurant usually leave tips (in a form of a currency note) for both of them in a tip jar. Given that there are *n* notes and every note have a positive value (not necessary the same value for each note) written on it. At the end of their workday, they arrange the notes arbitrarily from left to right on a table (this row of notes is not necessarily sorted). They play the following game to split the tip money: they take turns to play and at each turn, the player chooses either the leftmost note or the rightmost note and takes it. Justin is greedy and always plays using the following strategy: "If the leftmost note has value larger than the rightmost note, then take the leftmost note. Otherwise, take the rightmost note." Construct a dynamic programming algorithm that determines the plays for Randall such that the tip money he gets is *maximized*. Assume *n* is even and Randall plays the first turn. We also assume that the values for the notes are given to us—in order—form left to right in an input array v[1..n], i.e. v[1] is the value of the leftmost note and v[n] is the value of the rightmost note.

a) Define (in plain English) subproblems to be solved (5 pts)

Index the notes from leftmost to rightmost as $1, \ldots, n$. There corresponding values are $v_1, \ldots, v_n$. Define $OPT(i, j)$ as the maximum amount of money that Randall can get from the remaining note sequence $i, \ldots, j$.

b) Write the recurrence relation for subproblems (5 pts)

Since we know that Randall and Justin can pick a note at either end, we can derive 2 scenarios.

1). Randall picks a note on the left first. Next, Justin will either pick the $i+1$-th note or j-th note. We know that Justin will pick a note which will leave Randall with a minimum of the two. Therefore, we can say that Randall will get

a) if v(i+1)>v(j), $v_i + OPT(i+2, j)$

b) Otherwise, $v_i + OPT(i+1, j-1)$

2). Randall picks a note on the right first. Justin will pick the i-th note or the $j-1$-th note. Similarly, Randall will get

c) if v(j)>v(j-1), $v_j + OPT(i+1, j-1)$

d) Otherwise, $v_j + OPT(i, j-2)$

Therefore, we need to decide whether Randall should pick the i-th or j-th note first. Hence, we have the following recurrence equation,

$$OPT(i, j) = max\{above\ two\ cases\}$$

Rubric:

a), b), c), d) each worth one point

OPT(i, j) = max{above two cases} worth one point

Partial point:

+2: If you only mentioned 4 items, v_i + OPT(i+2, j), v_i + OPT(i+1, j-1), v_j + OPT(i+1, j-1), v_j + OPT(i, j-2) and without any analysis.

c) Using the recurrence formula in part b, write pseudocode to compute the maximum amount of tip for Randall. Do not forget the initialization step

```
Base case:
Assume OPT(1...n, 1...n) array initialized with 0.
Opt(i,i) = v[i], for i= 1,2,…,n
OPT(i,i+1) = max{V[i], V[i+1]} for i= 1,2,…,n-1
Find_max(V, n):
    for len from 2 to n, step_size = 2
      for i from 1 to n, j = len to n, step_size for both i and j is 1
          Recurrence function in b)
```

```
        return Opt(1, n)
```

Rubric:

Base case worth 1 point, as long as your answer contains OPT(i,i+1) = max{V[i], V[i+1]} for i= 1,2,…,n-1, you will get the point.

If your algorithm is a recursion version, you will only get 4 points if your algorithm is correct.

If your algorithm is an iteration version, there are two for loops, each worth 2 points.

-1 point: Missed return optimal value