**CS570 Spring 2023: Analysis of Algorithms Exam I**

|  | Points |  | Points |
| --- | --- | --- | --- |
| Problem 1 | 12 | Problem 5 | 16 |
| Problem 2 | 9 | Problem 6 | 18 |
| Problem 3 | 9 | Problem 7 | 24 |
| Problem 4 | 12 |  |  |
|  | Total | 100 |  |

Instructions:

1. This is a 2-hr exam. Closed book and notes. No electronic devices or internet access.
2. A single 8.5X11 cheat sheet is allowed.
3. If a description to an algorithm or a proof is required, please limit your description or

proof to within 150 words, preferably not exceeding the space allotted for that question.

1. No space other than the pages in the exam booklet will be scanned for grading.
2. If you require an additional page for a question, you can use the extra page provided

within this booklet. However please indicate clearly that you are continuing the solution on the additional page.

1. Do not detach any sheets from the booklet. Detached sheets will not be scanned.
2. If using a pencil to write the answers, make sure you apply enough pressure, so your

answers are readable in the scanned copy of your exam.

1. Do not write your answers in cursive scripts.
2. This exam is printed double sided. Check and use the back of each page.
3. 12 pts  
   Mark the following statements as **TRUE** or **FALSE** by circling the correct answer. No need to provide any justification.

**[ TRUE/FALSE ]**

A dynamic programming algorithm with a worst-case runtime of O(*n2*) takes O(*n2*) space.

**[ TRUE/FALSE ]**

The Bellman-Ford algorithm, when used for finding the shortest path in a directed acyclic graph (DAG) with negative-weight edges, is guaranteed to correctly find a shortest path.

**[ TRUE/FALSE ]**

When inserting a new element into a binary max-heap, if the new element is larger than all existing elements in the max-heap, insertion will take O(1) time.

**[ TRUE/FALSE ]**

If = and = , Then = .

**[ TRUE/FALSE ]**The top down pass in dynamic programming produces the **value** of the optimal solution whereas the bottom up pass produces the actual solution.

**[ TRUE/FALSE ]**

In a binary min-heap, the element with the maximum key value can be found in O(log n) time.

1. 9 pts (3 pts each. Select ALL correct choices. No partial credit)

I. After the Gale-Shapley algorithm has been run (between n men and n women with men proposing) and a stable match has been found, one man announces that his preference list was in reverse order, and that the algorithm needs to be re-run. If the Gale-Shapley algorithm is run again (with men proposing) with his preference order reversed, what is the minimum number of pairings that would change from re-running the algorithm?

* 1. *n*
  2. *n/2*
  3. *2*
  4. 1
  5. 0

1. Which of the following techniques and algorithms can be classified as divide and conquer?

* Binary search
* Bellman-Ford running in a distributed environment as described in class
* Kruskal’s algorithm
* Space efficient version of sequence alignment as described in class

1. To which of the following recurrence relations can the Master’s theorem be directly applied:
2. 9 pts  
   Arrange the functions below in order of increasing running time according to Big-*O* notation using = (equivalent) and ⊂ (strict subset of).

Assume all logs are based 2.

*Points: There will be a one-point deduction for each inversion in your ordered list*

1. lg (n2)
2. lg (n lgn)
3. 2lgn
4. 2n
5. lg (n2n)
6. 2n2
7. n lgn

Solution:

lg n2< lg (n lgn )< < 2lgn < lg n2n < 2n2 < n lgn < 2n

Rubric: -1 point per inversion

1. 12 pts  
   Given the undirected graph shown below:

A picture containing diagram, line, circle, drawing

Description automatically generated

1. Use Prim’s algorithm to obtain an MST of this graph. Use A as your starting point. Show the final MST and indicate the order in which you selected the edges

MST : (A,C), (A,B), (B,D), (D,H), (D,G), (G,F), (F,E)

* (B,D) could also be (C,D) or (C,H)
* Rubric: -1 point for each wrong step

1. Use Kruskal’s algorithm to obtain an MST. Show the final MST and indicate the order in which you selected the edges

MST: (G,F), (E,F), (A,C), (A,B), (D,H), (B,D),(D,G)

(E,F) can be swapped with (A,C), similarly (D,H) with (A,B), and (B,D) with (D,G).

(B,D) could be replaced by (C,D) or (C,H)

Rubric: -1 point for each wrong step

1. Is the MST in this graph unique? If yes, explain why. If no, show all edges that can be part of an MST but don’t have to be part of every MST.

No.

(B,D) , (C,D) , and (C,H)

Rubric: 1 point for answering ‘No’, 3 points for showing the 3 edges in the answer, -1 for each wrong.

1. 16 pts  
   Alice has a knapsack with capacity C that she will fill exactly to capacity with her choice of items from k item types. Each item type has a weight w\_i and value v\_i. Alice has an unlimited supply of each item type, in other words, Alice may choose more than one item of the same type. Design a DP algorithm to determine the worst case — minimum value — of items that will exactly fill up the knapsack and not leave any remaining space. You can assume that such a combination of items exists.
2. Define (in plain English) the subproblems to be solved. (4 pts)

Let OPT[i, j] = min value of when choosing from first i items with total weight exactly j

Rubric:

* -1 point for not describing that subproblems solve for the minimum value
* -1 point for incorrectly describing what i represents
* -1 point for incorrectly describing what j represents
* -4 points if definition missing or entirely incorrect

1. Write a recurrence relation for the subproblems (4 pts)

If w[i] <= remaining capacity of the knapsack

OPT[i, j] = min{OPT[i-1, j], v\_i + OPT[i, j - w\_i]}

Else:

OPT[i, j] = OPT[i-1, j]

Rubric:

* -1 point if not taking the min of the two cases
* 1.5 for each of the two cases (i.e. include or not include item i)
* -1 if not distinguished the case when w\_i > j (no penalty if handled via additional base cases)

1. (6 pts for part c)  
   Using the recurrence formula in part b, write pseudocode using iteration to find the minimum total value of items that can fill up the knapsack (4 pts)

Make sure you specify base cases and their values (2 pts)

Initialize OPT[0, j] = infinity For j from 1 to C (i.e. minimum value of knapsack when there are 0 item types and j > 0)

Initialize OPT[i, 0] = 0 For i from 1 to k (i.e. minimum value of knapsack when there is zero capacity in the knapsack)

For i from 1 to k

For j from 0 to C

If w[i] <= j:

OPT[i, j] = min{OPT[i-1,j], v[i] + OPT[i, j-w[i]]}

Else:

OPT[i, j] = OPT[i-1, j]

Return OPT[k,C]

Rubric:

* 1 point for each base case
* 1 point for overall correct structure (Base cases followed by double loops computing the recurrence)
* 1 point for each loop having correct

1. What is the time complexity of your solution? Is your algorithm efficient? (2 pts)

Solution takes O(kC)  
Not efficient. This algorithm is pseudo polynomial in C

6) 18 pts  
Given a unsorted list of *n* positive integers {*a*1, *a*2, … , *an*}, design a divide-and-conquer-based algorithm that runs in *O*(*n*) time and finds the maximum value of *ai* / *aj* where *i* < *j*. No proof of correctness is required, but analyze the time complexity of your algorithm.

(You will only get 50% partial credit if your divide and conquer algorithm has polynomial runtime but is not *O*(*n*).)

This is similar to the stocks buying-selling problem

1. Split the array into two equal halves L and R
2. Solve the problem recursively for each half (for cases that ai, aj are both in the left or right half (case 1 and 2 in the rubric)
3. For the case when i is from the first half, and j is from the second half: The ratio will be maximized when and . (case 3 in the rubric)

If we find these maximum and minimum elements in linear time by scanning the left and right parts, the complexity of the resulting algorithm will have recurrence T(n) = 2T(n/2) + O(n) which leads to T(n) = O(n log n).

To speed up, we can compute the minimum and maximum elements in a divide and conquer manner parallel to our main task (as shown by the code below):

1. Take the max of the 3 cases and return
2. Base case where we don’t divide further, can be array of size 1.

A picture containing text, screenshot, font, line

Description automatically generated

The complexity is then given by T(n) = 2T(n/2) + O(1) , thus, is O(n).

Rubric:

* 12 pts - correct algorithm
  + 4 pts - Compute best ai/aj in each half (case 1 and 2)
  + 4 pts - Considered case 3, i.e. Max-left / Min-right
  + 2 pts - Base cases
  + 2 pts - Complexity analysis
* 6 pts - Optimization with computing min\_left, max\_right in the same Divide-n-Conquer execution

(Refer to Gradescope for the rubric details on partial credit in the respective cases wherever applicable.)

7) 24 pts  
Suppose we have a group of students (evenly many of them) who need to be divided into pairs so that they can study in pairs for the CS570 exam. The students live in various locations around campus which are known. Due to commuting logistics, each student’s order of preference of who they want to be paired with, is directly governed by how far apart they live, i.e., any student i prefers student j over k if d(i, j) < d(i, k). For simplicity, assume the locations are such that all the pairwise distances are unique.

Consider the following algorithm to form the pairs:

Greedy-pair (S):

1. If S only has two students, pair them and end. If not,
2. Identify the pair of students i,j in S that live the closest among all possible pairs (remember such a pair is unique due to all distances being unique). Match i with j.
3. Remove i,j from S and repeat, i.e., execute Greedy-pair(S \ {i,j})

Your goal is to determine if this algorithm is good for two possible objectives:

1. Suppose the goal is to form pairs so as to minimize the sum of pairwise distances over all the pairs. Prove or disprove: The algorithm Greedy-pair(S) forms pairs so that it minimizes the total sum of pairwise distances. (12 points)

a) False. Construct a precise counter-example with a small instance where the algorithm doesn't work, such as: students a, b, c, d on a line with distances say

a —- 10 —- b —-1—- c —- 5 —- d (Making ac = 11, bd =6, ad = 16).

Then the algorithm pairs (b,c) and then (a,d) leading to a total distance of 1+16=17, whereas the minimum is 15 by pairing (a,b) and (c,d).

Rubric:

* 0 points: Trying to prove True
* 3 points: Answered false with no justification
* 6 points: Answered false but no/incorrect counter-example
* 9 points: Counter-example has minor errors
* 12 points: Correct Counter-example

(For both parts a and b below, refer to Gradescope for examples of major and minor errors)

See part b on next page

1. For given pairs formed, we say there is an instability due to any pairs (i,k) and (j,l), if i prefers j over k, and j prefers i over l. Prove or disprove: The algorithm Greedy-pair(S) forms pairs without any instabilities. (12 points)

b) True. If there was an instability due to i, j where (i,k) and (j,l) are the pairs, that would mean d(i,j) is smaller than d(i,k) as well as d(j,l). Suppose (i,k) was formed before (j,l), then, at that point, j was available with d(i,j) < d(i,k). Thus, the algorithm would not have picked (i,k) as the pair at that point, a contradiction. The same argument applies if (j,l) was formed before (i,k). Thus, there are no instabilities.

Rubric:

* 0 points: Trying to prove False
* 3 points: Answered True but justification missing or completely incorrect
* 6 points: Justification has major errors
* 9 points: Justification has minor errors
* 12 points: Correct justification

Additional Space

Additional Space