Monsoon 2024 Date: **09 December**

Graduate Algorithms: End-sem

Total: 70 points *Duration: 180 mins*

With every pseudocode, always include an explanation in words.
Common algorithms like binary search, sorting, graph algorithms done in course can be used directly.
You should ignore the COx tags against each question. Those are for administrative purposes.

Part-A (10 points)

Q1 [2x5 = 10 points] These questions do not require detailed answers; give 2-3 line answers. You cannot write "I don't know" for these questions. Assume $P \neq NP$, if required.

1. (CO3) A student claimed that polynomial-time many-one reductions exist only between NP-complete problems, and in particular, not between polynomial-time problems. Is she correct? Explain briefly.

2. (CO1) Consider n-element knapsack instances where the weight limit $W = \Theta(2^n)$. We want to understand the space complexity of the $O(nW)$ dynamic programming solution for two tasks. (a) Task-1: Compute only the optimal value. (b) Task-2: Compute the optimal value and the optimal set of items. Which of the tasks can be solved using space $poly(n)$ and why? If any of them can't be solved within the above limit, state which one and why. Explain briefly.

3. (CO2) Recall that the Floyd-Warshall's algorithm for APSP uses a function/subproblem $SP(u, v, k)$: $SP(u, v, k)$ is defined as the weight of a shortest $u \rightsquigarrow v$ path such that the non-terminal vertices on that path belong to $\{1, 2, \ldots, k\}$. Write down the base case(s) for SP and a recursive definition of SP.

4. (CO1) Consider the following divide-and-conquer algorithm for the closest pair of points problem on 1D: Divide the points into two roughly equal-sized sets – those to the left of the median point and the rest. Then, recursively compute the optimal distance among the pairs of points in the left half and the optimal distance among the pairs of points in the right half. Further, compute the optimal distance among all pairs of points one of which belongs to the left half and the other belongs to the right half. The algorithm returns the minimum of all the three optimal distances. What can we say about the correctness of this algorithm? What can we say about the complexity of this algorithm? Briefly explain.

5. (CO2) Consider the Kosaraju-Sharir algorithm for finding the strongly-connected components of a graph. Recall that the algorithm calls DFS twice: first, on G, and then on G_{rev} (the reverse graph of G). Therefore, it is a linear-time algorithm. A student suggested this modification: first, call DFS on G_{rev}, and then call DFS on G (all other steps and details remain unchanged). What can we say about the correctness of this algorithm? What can be we say about the complexity of this algorithm? Briefly explain.

Part-B (60 points)

10% if you write **"I don't know"** for any (sub)question.

Q2 [1+4+1+6=12 points] (CO3) The 2-PART optimization problem is defined on n-sized arrays with non-zero integers and its task is to partition the array into two disjoint subarrays, say P_0 and P_1, such that $\max(\sum_{x \in P_0} x, \sum_{x \in P_1} x)$ is as small as possible. For example, the optimum value for the array $[5, 10, 20, 30, 22]$ is 45 corresponding to the partition $[5, 10, 30], [20, 22]$. The 2-PART problem is NP-hard.

1. Define an NP-complete decision problem corresponding to the 2-PART optimization problem; write down its input and output. Call the decision problem 2-PART-DEC.

2. Prove that 2-PART-DEC is an NP problem. You need to prove all the necessary steps, including the correctness claims and complexity analysis; however, you can skip the proofs of the claims.

3. Write the name of some NP-complete problem Q that you would use to prove that 2-PART-DEC is NP-complete; write down the input and the output of Q.

4. Prove that 2-PART-DEC is NP-hard by describing a reduction involving Q. You need to provide all the necessary steps, including the correctness claims and complexity analysis; however, you can skip the proofs of the correctness claims.

Q3 [2 x 7 = 14 points] (CO4) Read the earlier question about the 2-PART optimization problem. In this question you will prove that the following algorithm yields a 3/2-relative approximation ratio.

```
def GREEDY2PART(X[1...n]):
1.  a ← 0 // running sum of the first partition P0
2.  b ← 0 // running sum of the second partition P1
3.  for i ← 1 to n
4.      if a < b // if the first partition P0 has a smaller sum right now
5.          a ← a + X[i] // add X[i] to P0
6.      else // if the first partition P0 does not have a smaller sum right now
7.          b ← b + X[i] // add X[i] to the P1
8.  return max(a,b)
```

1. Let the value of a and b after the (j)-th iteration of the for loop be denoted a_j and b_j, respectively (thus, $a_0 = b_0 = 0$). Run the greedy algorithm on $X = [5, 10, 20, 30, 22]$ and write the following: (i) values of $a_1, a_2, \ldots a_5, b_1, b_2, \ldots b_5$, and (ii) the return value of the algorithm.

2. Suppose OPT denotes the optimal value corresponding to some instance, and APPROX denotes the output of the greedy algorithm for that instance. If we want to prove that the approximation ratio of the algorithm is 3/2, how should OPT and APPROX be related?

3. Let $T = \sum_{x \in X} x$ denote the total sum of all elements in the input array. Relate OPT and T in the best possible manner. Briefly justify the relation.

4. Let $X_{max} = \max_{x \in X} x$ denote the largest element in X. Relate OPT and X_{max} in the best possible manner. Briefly justify the relation.

5. Suppose the greedy algorithm returns the value of the first partition P_0 at the end (the other case is handled in exactly the same manner). Further, suppose the j-th element of X, denoted X_j, is the last element that was added to P_0 by the greedy algorithm. Note that since P_0 was chosen (in line 4) for $X[j]$, $a_{j-1} < b_{j-1}$ must be true. Express APPROX in terms of a_{j-1} and $X[j]$. Briefly justify the relation.

6. Relate T and a_{j-1} in the best possible manner. Briefly justify how your arrived at the the relation.

7. Prove that $APPROX < T/2 + X_{max} \leq 2 * OPT$. Briefly justify how your arrived at the relation. Do not worry about the tightness of the inequalities; it is okay to prove $\leq$ for the first inequality and $<$ for the second inequality.
   Hint: You may not be able to derive this relation if you made a mistake in the relations above.

8. (Extra credit: 4 points, "I don't know" is not applicable) Relate T, a_{j-1} and $X[j]$ in the best possible manner. Briefly justify how your arrived at the the relation. Use this relation to prove that the greedy algorithm outputs a 3/2-approximate value.

Q4 [1 + 5 + 1 + 1 = 8 points] (CO1) Given a rooted binary tree T whose nodes are weighted and an integer S, consider a decision problem whose task is to find if T contains a node whose subtree has weight S. The subtree of a node consists of all descendants of that node, including itself; the weight of a subtree is the sum total weights of all such descendants. The weight of a node x can be retrieved by $wt(x)$; weights are non-zero.

1. Construct an instance (T^*, S^*) such that T^* contains a non-root node whose subtree has weight S^* but the root node of T^* does not have weight S^*. Mark that node, say x. State the weight of the subtree of x and the weight of T^*. Drawing a small tree may be helpful later.
2. Write down a divide-and-conquer algorithm to solve this decision problem. Remember to clearly describe its input(s) and output(s). Write both pseudocode and explanation of the code. Also, analyse the time-complexity of your algorithm.
3. Trace the algorithm on the above instance T^* and S^*; it is sufficient to show each recursive call, along with the corresponding input and output.
4. Suppose we want to prove the correctness of your algorithm. What induction hypothesis should be used? Only state the hypothesis; proof is not required.

Q5 [2+3+5+1+1+1+1=14 points] (CO1) A team of painters has been hired to paint several advertising boards. Your algorithm is given an array $B[1 \ldots n]$, representing the boards, and k as input. Each element of B represents the area of a board. k numbers of painters are available to paint these boards. Consider that each unit area of a board takes 1 unit of time to paint.

The task is to assign each painter some boards such that they take the minimum time to paint all the boards. Once the board-to-painter assignment is done, all the painters simultaneously start working, and the job is done when all the boards are painted. The assignment must be made under the constraint that any painter will only paint a continuous sections of boards, say boards [2, 3, 4] or only board [1] or but not boards [2, 4, 5]. It is okay if some painter is not assigned any board.

For example, for $k = 3$ and $B = [5, 10, 30, 20, 15]$, the optimum time is 35 which can be achieved by assigning boards 1 and 2 to painter 1, 3 to painter 2, and 4, 5 to painter 3; job is done when all the boards are finished at time $\max(5 + 10, 30, 20 + 15) = 35$. Similarly, when $B = [10, 20, 30, 40]$ and $k = 2$, the optimal assignment is boards 1,2,3 to painter 1 and board 4 to painter 2; job will be completed at time $\max(10 + 20 + 30, 40) = 60$.

Design a dynamic programming solution for this problem. Do not write any pseudocode, instead, present your solution in this manner.

1. Definition of a function/subproblem; you have to specify the ranges of the parameters of the function.
2. Draw the memo for the instance $B = [5, 10, 30, 20, 15]$ and $k = 3$ corresponding to the function/subproblem that you defined in the previous question, and fill all entries of the memo.
3. Recursive formula for the function, including the base case(s).
   Hint: You should check that the values you computed above matches the formula you derive here.
4. How to solve the original question from the memo?
5. How to store and fill the memo?
6. Time complexity of filling the memo.
7. Space complexity of filling the memo.

Q6 [1+6+3+2=12] (CO2) You have been approached by a logistics company to help them with managing boxes while loading them in shipping containers. Boxes arrive on the shipyard one-by-one (assume that all of them have the same size for the sake of simplicity) and the workers stack them in some manner; the operations manager wants to keep track of the stack sizes to ensure safety and profitability. Essentially, you have to implement three subroutines:

- Newbox(id): a new box with identifier *id* has just arrived. New boxes are kept on the floor by itself and is not immediately stacked.
- Move(x,y): a worker just now lifted the stack containing box x and placed it directly on top of the stack containing box y. x and y are ids of two boxes that have already arrived.
- Load(x): should return the number of boxes in the stack containing box x. x is id of some box which has already arrived.

1. Consider the following sequence of operations: Newbox(1), Newbox(2), Move(1,2), Newbox(3), Newbox(4), Move(4,3), Move(2,4). How many stacks are there? Write down the boxes in each stack in a top-to-bottom manner.
2. Imagine that you have a union-find/disjoint-set library at your disposal. Implement all the three subroutines Newbox, Move, and Load by making Makeset, Union, and Find calls to a union-find data structure. Your implementation must match the API and functionality specified above.
   You can store additional fields in the set elements (recall: the least-common-ancestor algorithm that we studied in class stored an additional field "name" in the set elements). Your implementations should not use any other local or global data structure or any global variables. It should not make any non-standard call to the union-find data structure; further, do not assume anything about the implementation of the union-find data-structure. Write both pseudocode and its explanation.
3. Suppose the worst-case running times of Makeset, Union, and Find are denoted T_M, T_U, T_F, respectively. Analyse the running time of Newbox, Move, and Load at a time when there are n boxes in the shipyard.
4. Among the four implementations of the union-find data-structure that we studied, which one would you use and why?

I am glad we could complete this journey together! All the best.