Monsoon 2024 Graduate Algorithms: Mid-Sem Exam *Max: 90 points*
Due: **Tuesday, 8 October** *Duration: 120 mins*

Most questions demand a short justification, and for them it is mandatory to include a **brief explanation using 3-4 sentences** (anything more is liable to be ignored). Do not forget to include base cases for recurrences and recursive algorithms.

Read all the questions carefully before attempting. Their ordering does not reflect their difficulty.

If you do not wish to attempt a question, you may write "I don't know". You can do so for Q1—Q9 and each subquestion of Q8. You will be awarded 20% of the marks of that question (subject to an overall cap of 8 marks) for understanding no answer is better than a bad one.

Q1 [10 points,CO1] Prove that $T(i,j) = O((i+j)^2)$ is a solution to the following recurrence. Show the steps. You may use any method but you must ensure all the cases of the relation as given below are satisfied.

- $T(i,j) = \max\{T(i-1,j), T(i,j-1)\} + (i+j)$ for $i > 0, j > 0$,
- $T(i,0) = T(i-1,0) + i$ for $i > 0$,
- $T(0,j) = T(0,j-1) + j$ for $j > 0$,
- $T(0,0) = d$

Q2 [5 points,CO1] This question is about the median-of-median algorithm. Suppose we modified the algorithm to split the input array into blocks of 11 elements. Write down the recurrence for the time-complexity of *MOMSelect* and state its solution. Derivation is not required for full-credit.

You will get additional 5 BONUS points for showing the derivation of the solution. Your derivation must be completely correct to get those 5 points – there will be no partial marking for the BONUS points.

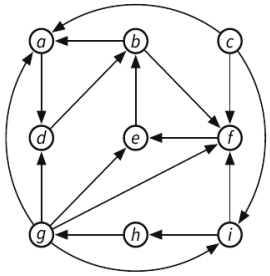
Q3 [5 points,CO1] A function $Foo(i,k)$ is defined for any $i \geq 0$ and $k \geq 0$ in the following manner.

$$Foo(i,k) = \begin{cases} k & \text{if } i = 0 \\ \max(0, k-i) & \text{if } k > n \\ \max\left\{ \begin{array}{c} i+k+Foo(i-1,k+1) \\ i+Foo(i-1,k) \\ k+Foo(i,k+1) \end{array} \right\} & \text{otherwise} \end{cases}$$

Suppose we want to compute $Foo(n,1)$. (a) Describe an appropriate memoization structure and (b) an evaluation order for filling the memo (you can also explain the order using for-loops). (c) Lastly, state the running time of the resulting iterative algorithm to compute the requested function value.

Q4 [5 points,CO2] Is it possible to have a graph with a negative-weight cycle in which the shortest-path distance between two vertices is well-defined? If no, explain why. If yes, draw a graph G with a negative-weight cycle and two vertices s and t such that the shortest path between s and t has a distance of 5; use at most 4 vertices for G and clearly mark the negative-weight cycle and the shortest path between s and t.

Q5 [5 points,CO2] Clearly indicate the following structures in the directed graph below, or write NONE if the indicated structure does not exist.



1. Strongly-connected components
2. Strongly-connected component graph

Q6 [10 points,CO2] Let G be a directed graph with arbitrary edge weights (which may be positive, negative, or zero) but without negative cycles, and let s be an arbitrary vertex of G.

A student was given G and he handed me a list of values $dist[v]$ for every vertex which he claims are the shortest-path distances from s to v.

(a) Design an algorithm (write both pseudoode and high-level explanation) that will allow me to quickly verify this fact. The algorithm should return True if $dist[v]$ = shortest-path distance from $s \rightsquigarrow v$ for all v, and False otherwise. If you want to reuse any algorithm from lecture (or known from earlier), you need to write its pseudocode too along with a brief explanation.

(b) Briefly explain why your approach is correct (you need not give a rigorous proof using induction, contradiction, etc.).

(c) Analyse the running time of your algorithm.

For full-credit your algorithm must run in linear time.

Q7 [10 points,CO1] This is a question about finding triangles which I have simplified to 1D for the exam. Consider a set of points Q on the number line (you can think of Q as a sorted array of integers– $Q[1]$ is the leftmost point). Define any three points q_1, q_2, q_3 as a triangle, say T, whose perimeter is defined as $P(T) = |q_1 - q_2| + |q_1 - q_3| + |q_2 - q_3|$. Design and analyse an efficient divide-and-conquer algorithm to identify a triangle with the smallest perimeter. For example, if the points are $-10, -3, -1, 5, 6, 11, 13, 20$, then your algorithm should return 12 corresponding to the triangle $5, 6, 11$.

You must state the API of your algorithm, briefly explain the idea behind its correctness (you need not give a rigorous proof using induction), and analyse its time-complexity with the help of a recurrence.

For full-credit, your algorithm must run in linear time.

Q8 [6+2+6+6=20 points, CO1] We are given a stack of n pancakes of different sizes. You can visually find out which pancake is larger than which one (and even the smallest and the largest ones), however, you are only allowed to insert a spatula below some pancake and "flip" the entire lot from the top to that pancake (as shown below).

(a) Design an efficient recursive algorithm `SortPancakes(...)` to in-place sort the entire stack of pancakes from the smallest diameter (at top) to the largest diameter (at bottom). Assume that there is a global array D of diameters ($D[1]$ represents the diameter of the topmost pancake). Design the algorithm such that after `SortPancakes(...)` has finished, D would be sorted; your implementation of `SortPancakes` can take input parameters and produce outputs – be sure to explain them in the API.

Your algorithm can call the following subroutines all of which take $O(1)$ time.

- $Flip(i)$ will flip the top-i pancakes. E.g., if current $D = [8, 1, 6, 9, 3]$, after $Flip(3)$, D will be $[6, 1, 8, 9, 3]$.
- $Min(k)$ will return the index of the minimum diameter of the topmost k pancakes.
- $Max(k)$ will return the index of the maximum diameter of the topmost k pancakes.

For full-credit in Q8, your algorithm should make at most $2n - 3$ flips. *Hint:*Base case!
For 80% partial credit, your algorithm should make at most $2n$ flips. No credit otherwise.

(b) Trace your algorithm for $D = [3, 1, 2]$. Clearly indicate the recursive calls.

(c) Prove that your algorithm is correct using induction. State the induction hypothesis, and then prove it (include the base case).

(c) Use a recurrence relation to analyse the precisely derive the worst-case number of "flips" used by your algorithm n (I am asking for a precise expression in n, and not an asymptotic function like $O(n)$).

Q9 [20 points,CO1] Let T be a rooted tree n nodes with integer weights on its edges, which could be positive, negative, or zero. The weight of a path in T is the sum of the weights of its edges; there should be at least one edge in a path. Describe and analyze a dynamic programming algorithm to compute the minimum weight of any path from *a node (not necessarily the root) down to one of its descendants (not necessarily a leafnode)*. It is not necessary to compute the actual minimum-weight path; just its weight is sufficient. For example, given the left tree shown below, your algorithm should return the number -12 (the minimum-weight downward path in this tree is shaded).

For full-credit your algorithm must run in linear time, and for partial-credit it must run in $O(n^3)$. Nothing otherwise.

(a) Write your solution according to the template shared in class.
(b) Create the memo for the right tree shown below; use the definition of the function/subproblem for computing the memo entries.