Monsoon 2025 Graduate Algorithms: Endsem Exam *Max: 105 points*
Date: **3 December** *Duration: 180 mins*

10% if you write **"I don't know"** for any (sub)question with a cap of 10 points.

Q1 [45 points] These questions are based on the lectures and homeworks.

(i) [5 points] Consider the MomSelect algorithm. One student made a mistake while implementing it. Instead of returning the medians of the 5-element arrays, he returned their 4th smallest elements (median would be the 3rd smallest element). Write down the recurrence for the time-complexity of this algorithm.

(ii) [5+2=7 points] Suppose we want to multiply two polynomials $P(x) = 1-2x+3x^2+4x^3$ and $Q(x) = 6-2x^3$ using the DFT-IDFT approach: compute DFT of both $P(x)$ and $Q(x)$, take their pointwise products, and take IDFT of the resulting array to output a polynomial $R(x)$. Suppose a student decided to use DFT and IDFT of order 4.

(a) Write down the DFT of $P(x)$. You need not show any steps, you need not even run the FFT algorithm; write only the answer as complex numbers of the form "a+ib".

(b) Let $PQ(x)$ denote the product $P(x) * Q(x)$. Is $PQ(x) = R(x)$ for all x, for some x, for no x? Briefly explain. You do not need to write $PQ(x)$ or $R(x)$.

(iii) [4+1=5 points] Call a sequence $X[1 \dots m]$ oscillating if either one of these holds:

- $X[i] < X[i+1]$ for all even i, and $X[i] > X[i+1]$ for all odd i.
- $X[i] < X[i+1]$ for all odd i, and $X[i] > X[i+1]$ for all even i.

Let $A[1 \dots n]$ be a sequence of integers. Define $sos(i, mode)$ for $i \in \{1, \dots n\}$ and $mode \in \{" > ", " < "\}$ as the length of the shortest oscillating supersequence of $A[i \dots n]$ such that the first two elements of the supersequence are related according to $mode$. For example, $sos([1, 12, 23, 34], " < ") = 6$ (corresponding to 1,12,1,23,1,34) and $sos([1, 12, 1], " > ") = 4$ (corresponding to 10,1,12,1 or 1,0,12,1). Give (a) a recursive formula for $sos()$, (b) including the base cases. Explain how it performs an exhaustive case analysis.

(iv) [3+2=5 points] Let $Reduce1$ be the reduction from 3SAT to CLIQUE and $Reduce2$ be the reduction from CLIQUE to INDEPENDENT-SET. Consider this reduction from 3SAT to INDEPENDENT-SET: Given a 3SAT instance F on n variables in c clauses, it first computes $(G, k) \leftarrow Reduce1(F)$. Then, it computes $(H, t) \leftarrow Reduce2(G, k)$ and returns (H, t). (a) Then, what would be the number of vertices in G and what would be k? (b) Next, what would be the number of vertices in H and what would be t?

(v) [5+3=8 points] Given an undirected (not necessarily connected) graph $G = (V, E)$ with n vertices and m edges, and a list of pairs of vertices $S = [(u_1, v_1), (u_2, v_2), \dots (u_q, v_q)]$, (a) write the pseudocode of a disjoint-set based algorithm to print, for every pair (u_i, v_i) if u_i and v_i are in the same connected component. Your algorithm should not use any global variable or data structure other than a disjoint-set and should only use the Makeset, Find, and Union calls on that set. (b) Next, analyse the time-complexity of your algorithm if we use the reversed-tree + union-by-depth implementation of disjoint-set.

(vi) [2+2+2=6 points] Describe (a) the NP-complete Knapsack problem, and (b) the optimization version of the Knapsack problem (no algorithm, reduction, or example is required). (c) What is the optimal (smallest) approximation ratio for the optimization problem if every item has value between 0 to 1023? Explain.

(vii) [4 points] Why is this algorithm not a 2-approximation algorithm for vertex cover on connected graphs: From the input graph G, choose a vertex s and run breadth-first search. Let T denote the breadth-first search tree. Return the internal nodes of T.

(viii) [1+4=5 points] Write down the description (instance and question) of the CHROMATIC NP-complete problem. In class we saw why there cannot be a 1-absolute approximation algorithm to compute the chromatic number of a graph. Suppose you are allowed to call $ApproxCHR(G)$ — a 2-absolute approximation algorithm that runs in polynomial time. Write the pseudocode, along with brief justification of correctness, of an algorithm solving CHROMATIC in polynomial time.

Q2 [1+2+2+16+4=25 points] Several years after graduating from IIIT-Delhi, you decide to open a ~~one-day~~ pop-up art gallery selling NFTs, using the following dynamic pricing strategy. All NFTs art your gallery have the same advertised price, ~~which you set at the start of the day,~~ but which you can decrease later. Customers visit your gallery one at a time. If a customer is willing to pay your current advertised price, they buy one NFT at that price. On the other hand, if your advertised price is too high, the customer will suggest a lower price that they are willing to pay. If you refuse to lower your advertised price, the customer will leave without buying anything. If you agree to lower your advertised price to match their offer, the customer will buy one NFT at the new lower price. Whenever you lower your advertised price, your new lower price stays in effect until you lower it again, ~~or until the end of the day~~. You can never increase your advertised price. You know your customers extremely well, so you can accurately predict both when each customer will come to the gallery, and how much each customer is willing to pay for one of your NFTs. Assume that only one customer comes to your gallery on each day.

Your goal is to compute the maximum amount of money you can earn using this dynamic pricing strategy. Your input consists of an array $V[1 \ldots n]$, where $V[i]$ is the amount that the i-th customer (in chronological order), equivalently, the customer on the i-th day, is willing to pay for one NFT. For example, if the input array is $[5, 3, 1, 4, 2]$ (representing 5 days), your algorithm should output 13, because you can earn 5+3+0+3+2 = 13 dollars using the prices $[5, 3, 3, 3, 2]$, and this is optimal.

(a) What is the optimal sequence of prices and the optimal earning for input array [6,2,3]?

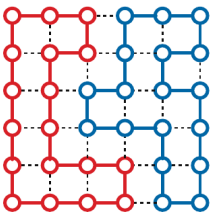
(b) Consider the following greedy approach: advertise the price $V[1]$ on the first day, and for each subsequent day, lower the price only if required to match an even lower customer's suggestion. Construct an instance on 2 days to show that this algorithm does not return the optimal solution. Write the optimal earning, the sequence of prices returned by the greedy approach and the corresponding earning.

(c) Suppose $n = 2$, i.e., you will open your gallery only for 2 days. Explain how to determine the optimal sequence of prices.

(d) Design a dynamic programming approach to determine the maximum earning for any n.

(e) Draw the memo corresponding to (d) for $V = [6, 2, 3]$.

Q3 [6+9=15 points] A Hamiltonian bicycle in a graph G is a pair of simple cycles in G, with identical lengths, such that every vertex of G lies on exactly one of the two cycles; a simple cycle is one that does not visit a vertex twice. The figure below show a Hamiltonian bicycle in the 6x6 grid graph.



Prove that it is NP-complete to determine whether a given graph G has a Hamiltonian bicycle. Write down (a) a verification algorithm and (b) a reduction algorithm. You have to write down the claims for the correctness of your verifier and reduction, but you do not need to prove them.

Q4 [10+5+5=20 points] Let G be a connected undirected graph. Suppose we start with two coins on two arbitrarily chosen vertices of G, and we want to move the coins so that they lie on the same vertex using as few moves as possible. At every step, each coin must move to an adjacent vertex; it is allowed for any coin to land on a vertex multiple times and it is also allowed for both coins to be on the same vertex.

Describe and analyze an algorithm using the graph reduction technique to compute the minimum number of steps to reach a configuration where both coins are on the same vertex, or to report correctly that no such configuration is reachable. The input to your algorithm consists of a graph $G = (V, E)$ and two vertices $u, v \in V$ (which may or may not be distinct). For full credit, your algorithm should run in $O(V + E)$ complexity; any slower but polynomial-time approach will fetch at most 15.

Do **not** write a pseudocode, instead, (a) describe the vertices and edge of another graph H, (b) then, explain what problem will you solve on H and what algorithm will you choose, and (c) finally, analyse the complexity of your approach in terms of V and E, the number of edges and vertices of the original graph G, respectively (the complexity should include the time to construct H and the complexity of the algorithm on H).