Winter 2023
May 4, 2023

ADA Endsem Rubrics

Max: 60 Marks
120 minutes

Roll No: _____ **Section:** __

Name: _____

Please write solutions independent of each other. This is a cheat based examination. You can carry **two A4-sized reference sheets** *where you can have some formulas etc written. Those two A4-sized reference sheets must contain your roll number and section and they have to be submitted along with the answer booklet. Moreover, those cheat-sheets cannot be shared with others. Furthermore, your solution must fit in the space provided. Extra sheet will be provided* only *for rough work . So try to be precise and brief. Meaningless blabber fetches negative credit. Also, you can use anything as a subroutine that was taught in the lectures or tutorials.*

Part A

Question	1	2	3	4	5	6	7	8	9	10	Total
Marks											

Part B

Question	1	2	3	4	Total
Marks					

Total Marks:

Part A

This part has 10 questions. Answer any 8 questions.

1. **(3 Marks)** Let G be a directed graph having $s, t \in V(G)$ and positive edge weights. Consider a longest simple path P (i.e. a path P with maximum total edge weight) from s to t in G. Let $u \in P$ be an internal vertex in this path P. Your friend claims that the subpath of P from s to u is a longest simple path from s to u in G. Is this claim **TRUE** or **FALSE**? You do not need to justify your answer.

   Answer : FALSE.

   Consider the graph with the following edges $(s, 1)$, $(s, 2)$, $(1, 4)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(4, t)$. The longest path form s to t is $s - 2 - 3 - 4 - t$. Longest path form s to 2 is $s - 1 - 4 - 2$.

   Rubric : +3 for correct / **0** for incorrect.

2. **(3 Marks)** Which of the following statements (**it could be more than one**) is/are **TRUE** for an undirected graph?
   Statement P: Number of odd degree vertices is even.
   Statement Q: Sum of degrees of all vertices is even

   (a) Only statement P.

   (b) Only statement Q.

   (c) Both statements P and Q.

   (d) Neither statement P nor statement Q.

   Answer : (c)

   Q is true: Since the graph is undirected, every edge increases the sum of degrees by 2. P is true: If we consider sum of degrees and subtract all even degrees, we get an even number (because Q is true). So total number of odd degree vertices must be even.

   Rubric : +1 for (a) / **+1** for (b) / **+3** for (c) / **0** for (d)

3. **(3 Marks)** Consider an edge weighted undirected connected graph G such that all edge weights are positive and distinct. Let $e_{\min}$ and $e_{\max}$ be the edges with smallest weight and largest weight in G respectively. Then which of the following statements is **FALSE**?

   (a) Every minimum spanning tree of G must contain $e_{\min}$.

   (b) If $e_{\max}$ is in a minimum spanning tree, then its removal must disconnect G.

   (c) No minimum spanning tree contains $e_{\max}$.

   (d) G has a unique minimum spanning tree.

   Answer : (c)

   As edge weights are unique, there will be only one edge $e_{\min}$ and that will be added to MST, therefore option (A) is always true. As spanning tree has minimum number of edges, removal of any edge will disconnect the graph. Therefore, option (B) is also true. As all edge weights are distinct, G will have a unique minimum spanning tree. So, option (D) is correct. Option C is false as $e_{\max}$ can be part of MST if other edges with lesser weights are creating cycle and number of edges before adding $e_{\max}$ is less than $(n - 1)$.

   Rubric : +3 for (c) / **0** for others

4. **(3 Marks)** What is the time complexity of the following recurrence relation? $T(n) = 8T(\sqrt{n}) + (\log n)^2$? The base case of this recurrence relation is $T(n) = 1$ for $n \leq 2$. Give the tightest possible $f(n)$ in Big-Oh notation. Please write down the answer only. An explanation is not needed.

   Answer : $T(n) = \theta(\log_2 n)^3$

   Given $T(n) = 8T(\sqrt{n}) + (\log n)^2$. Let's take $n = 2^m$ Then , $T(2^m) = 8T(2^{m/2}) + m^2$. Let's take $T(2^m)$ as $S(m)$, $S(m) = 8S(m/2) + m^2$. Apply Master's Theorem, $a = 8, b = 2, d = 2$. By case: $d < \log_b a$, $S(m) = O(m^3)$. Put m as $\log_2 n$. Thus, $T(n) = O(\log_2 n)^3$.

Rubric : Any process of proving is acceptable. Steps should be clear and correct. **+1** for any correctly proven loose bound give. **0** for incorrect steps and loose bound give. **+2** for minor mistakes and loose bound give. **+3** clear steps and tight bound. **0** very unclear steps or incorrect bound give.

5. (3 **Marks**) There are five students in a class namely X, Y, Z, W, and V. Student X claims that a triangle is a bipartite graph. Student Y claims that a pentagon is a bipartite graph. Student Z claims that a square is a bipartite graph. W claims heptagon is a bipartite graph. Student V claims that a decagon is a bipartite graph. Who among them is/are correct? (for instance, if you think X and Z are saying correct but not the rest, then you should write "only X and Z are saying correct statement")

   Answer : Only V and Z are saying the correct statement.

   The argument follows from the fact that even length cycles are bipartite graph, because they can be colored by two colors. But for odd length cycles, we require 3 colors, thus not bipartite graph.

   Rubric : +1 for one correct and one incorrect answer / **+2** for two correct and one incorrect answer /**+2** for one correct answer and no incorrect answer / **+3** for all correct answers and no incorrect answer / **0** for more than one incorrect answer.

6. (3 **Marks**) Let G be an undirected graph. Consider a depth-first traversal of G starting from s and let T be the resulting depth-first search tree rooted at s. Let u be a vertex in G and let v be the first new (unvisited) vertex visited after visiting u in the traversal. Which of the following statements is always **TRUE**?

   (a) (u, v) must be an edge in G, and u is a descendant of v in T.

   (b) (u, v) must be an edge in G, and v is a descendant of u in T.

   (c) If (u, v) is not an edge in G then u is a leaf in T.

   (d) If (u, v) is not an edge in G then u and v must have the same parent in T.

   Answer : (c)

   In DFS, if v is visited after u, then either (u, v) is an edge or u is a leaf node. In DFS after visiting a node, we first recur for all unvisited children. If there are no unvisited children (u is leaf), then control goes back to parent and parent then visits next unvisited children.

   Rubric : +3 for correct / **0** for incorrect.

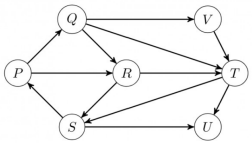
7. (3 **Marks**) Every directed acyclic graph has exactly one topological ordering. Is this **TRUE** or **FALSE**? Explain.

   Answer : FALSE.

   Some priority constraints may be unspecified, and multiple orderings may be possible for a given DAG. For example a graph $G = (V, E) = (\{a, b, c\}, \{(a, b), (a, c)\})$ has valid topological orderings $[a, b, c]$ or $[a, c, b]$.

   Rubric : +3 for correct / **0** for incorrect.

8. (3 **Marks**) Which of the following is the correct decomposition of the directed graph given below into its strongly connected components?



   (a) {P, Q, R, S, T, U, V}

   (b) {P, Q, R, S, T, V}; {U}

   (c) {P, Q, S, T, V}; {R}; {U}

   (d) {P, Q, R, S}; {T}; {U}; {V}

   Answer : (b)

A graph is said to be strongly connected if every vertex is reachable from every other vertex. The strongly connected component is always maximal that is if x is strongly connected component there should not exist another strongly connected component which contains x. If we take R as a strongly connected component but which is part of $PQRS$ and $PQRS$ is part of $PQRSVT$.

Rubric : +3 for correct / **0** for incorrect.

9. (3 **Marks**) Consider a simple undirected weighted graph G all of whose edge weights are distinct. Which of the following statements about the minimum spanning trees of G is/are **TRUE**?

   (a) The edge with the second smallest weight is always part of any minimum spanning tree of G.

   (b) One or both of the edges with the third smallest and the fourth smallest weights are part of any minimum spanning tree of G.

   (c) Suppose $S \subseteq V$ be such that $S \neq \varnothing$ and $S \neq V$. Consider the edge with the minimum weight such that one of its vertices is in S and the other in $V \setminus S$. Such an edge will always be part of any minimum spanning tree of G.

   (d) G can have multiple minimum spanning trees.

   Answer : (a), (b) and (c)

   Rubric : 0 for any incorrect answer / **+1** for one correct answer / **+2** for two correct answer / **+2** for all correct answers

10. (3 **Marks**) Show: $O(\log_2 N) = O(\log_7 N)$.

    Answer : This is an exercise of changing base. We know,

$$\log_a b = \frac{\log_c b}{\log_c a}$$

So we can write,

$$\log_7 N = \frac{\log_2 N}{\log_2 7}$$

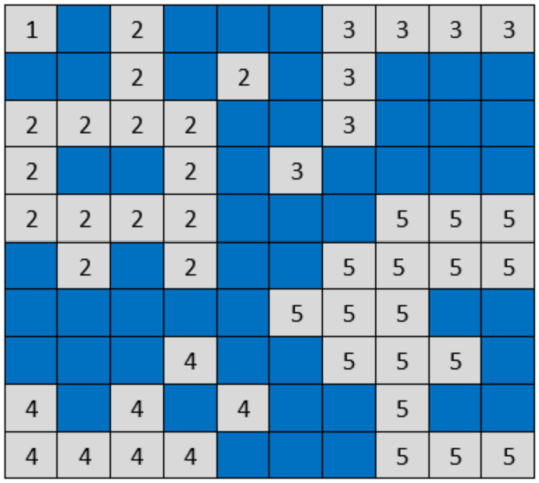
Since $\log_2 7$ is a constant, we have $O(\log_2 N) = O(\log_7 N)$

Rubric : +3 for correct proof / **+1** for partial steps or directly writing $\log_7 N$ is constant factor multiplicative of $\log_2 N$ / **0** for incorrect proof

Part B

This part has 4 questions. Answer any 3 questions.

1. **(12 Marks)** Given an $m \times n$ binary matrix whose 0 represents water and 1 represents land. The connected 1s form an island. Formally, a cell containing 1 with indices (i, j) (i indicates row and j indicates column) is considered connected to either of the following cells if that cell contains 1: $(i-1, j)$, $(i-1, j-1)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j)$, $(i-1, j+1)$, $(i+1, j-1)$, and $(i+1, j+1)$. The below figure depicts a problem instance. Here the light-colored cells have 1s, and dark-colored cells have 0s. There are a total of 5 islands as indicated by digits 1 - 5. Design an $O(mn)$-time algorithm to count the number of islands.



Graph construction:

(**+2** for identifying the problem is to find connected component) : The solution is inspired by finding the total number of connected components in a graph problem. The idea is to start Breadth–first search (BFS) from each unprocessed node and increment the island count. Each BFS traversal will mark all cells which make one island as processed. So, the problem reduces to finding the total number of BFS calls.

(**+3** for identifying the proper vertices and edges corresponding to matrix) : Before BFS we have to convert the given matrix in to graph. We consider each cell marked 1 as vertex and and put an edge between each neighbouring cells if the neighbour is marked 1. A cell has at-most 8 neighbouring cells and at-least 3 neighbouring cells. Thus the complexity of the graph is bounded by $m \times n$.

(**+4** for showing the construction, specifically identifying the 8 adjacent cells for edges) : Let I be the given matrix, So the construction for graph G is as follows:

(a) $G(V) \leftarrow \{\phi\}$; $G(E) \leftarrow \{\phi\}$

(b) For each (i, j) pair
If $I_{i,j} = 1$ then $V \leftarrow V \cup \{v_{i,j}\}$

(c) For each $v_{i,j} \in V$
If $v_{i-1,j-1} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i-1,j-1}\}$
Else if $v_{i-1,j} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i-1,j}\}$
Else if $v_{i-1,j+1} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i-1,j+1}\}$
Else if $v_{i,j-1} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i,j-1}\}$
Else if $v_{i,j+1} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i,j+1}\}$
Else if $v_{i+1,j-1} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i+1,j-1}\}$
Else if $v_{i+1,j} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i+1,j}\}$
Else if $v_{i+1,j+1} \in V$ then $E \leftarrow E \cup \{v_{i,j}, v_{i+1,j+1}\}$

Now we are ready for the algorithm.

Steps of the algorithm:

(**+3** for correctly computing number of connected components, with the help of a counter)

(a) Initialize an empty queue Q. Also set a counter C to be 0.

(b) While doing BFS on G using Q

- If Q is empty
  $C \leftarrow C + 1$

(c) Output C

2. (12 **Marks**) Given an $m \times n$ matrix of characters, find the length of the longest path in the matrix starting from a given character. All characters in the longest path should be increasing and consecutive to each other in alphabetical order. We are allowed to search the string in all eight possible directions, i.e., North, West, South, East, North-East, North-West, South-East, South-West. For example, consider the following matrix of characters:

```
[ D, E, H, X, B ]
[ A, O, G, P, E ]
[ D, D, C, F, D ]
[ E, B, E, A, S ]
[ C, D, Y, E, N ]
```

The length of the longest path with consecutive characters starting from character C is 6. The longest path is: C(2, 2) → D(2, 1) → E(3, 2) → F(2, 3) → G(1, 2) → H(0, 2)

[**Hint:** DP on DAG]

As the hint suggested, we can view the problem as finding the longest the longest path form a given node in the DAG.

The creation of DAG is straight forward. For each cell (i, j) of the matrix we consider a vertex $v_{i,j}$ in our graph like the last question. Now for each cell we check all of it's neighbouring cells. We put a directed edge from vertex $v_{i,j}$ to vertex $u_{k,l}$ if (k, l) is neighbour cell of (i, j) such that the character in (i, j) is smaller than the character in (k, l) in alphabetical order. Observe that the created graph say G is a DAG by definition as any path in the graph maintains alpabetical ordering. Now the problem is to find the longest path for a given node in a DAG.

Note that, if a neighbour of a cell contains the same character as the cell, we will put directed edge between them to any of the direction. This may effect the optimal solution, but this case is acceptable for scoring.

(**+4** for subproblem)

- Subproblem definition:.

  For a given vertex v let $ch(v)$ denotes children of v. $LP(v)$ denotes the longest path starting form v in the graph $\forall v \in V(G)$.

  (**+3** for recurrence)

- Recurrence Relation:

$$LP(v) = \max\{LP(v), 1 + \max_{u \in ch(v)}\{LP(u)\}\}$$

  Base case : $LP(u) = 0; \forall u \in V : ch(u) = \{\phi\}$

(**+2** for this section)

- Subproblem that solves the actual problem:
  For a given cell (i, j); $LP(v_{i,j})$ gives the actual solution.

(**+3** for algorithm)

- Description of the algorithm:
  (a) Create a graph as discussed above.
  (b) For a given cell (i, j), return $LP(v_{i,j})$

  LP(v)

  (a) If $ch(v) = \{\phi\}$
      – return 0
  (b) $len \leftarrow 0$
  (c) For each $u \in ch(v)$ then
      – $len \leftarrow \max(len, 1 + LP(u))$
  (d) return len

  Note : The problem can be solved without using the concept id DAG. The DP can be defined over the matrix directly. However, the solution sketch remains the same and instead of children, one can use the concept of adjacent cells directly. Then the direction of movement needs to be clearly defined as there is no directed edge defined on the matrix directly.

3. **(12 Marks)** There are n teams that plan to attend a grand dinner in a restaurant. Team i has t_i members. The restaurant can provide m tables for the dinner, with $m \geq n$. Table i can has s_i chairs. We wish to seat all the teams such that no two team members are at the same table, so that we maximum students getting to meet members of other teams. Can we do so?
Model this as a network flow problem. Specify the vertices, edges and capacities. Then finally, explain how you can transform the maximum flow into an optimal solution to this problem.

- Flow network construction (including vertices, edges, capacities):
(**+4** for proper construction of the graph. Any correct drawing of graph with little elaboration is also accepted) : Create a flow network with $n + m + 2$ vertices. Create one vertex for each team and one for each table. Create extra source and sink vertices. Create edges from the source to each team with a capacity of t_i. Create edges from each table vertex to the sink vertex with capacity s_i. Finally, add edges from each team to each table, with capacity 1, since each team can provide at most one person per table.

- How does the maximum-flow of your network can be transformed into an optimal solution to this problem?
(**+3** for the point. **0** if not mentioned correctly) : Run the network flow algorithm. If the maximal flow equals the sum of the number of team members, the seating can be done. Otherwise, it can not be.

- Algorithm Description:
  (**+2** for this section)
  (a) Create the graph as explained above
  (b) Use the Ford Fulkerson's algorithm to get the max-flow from s to t for the above graph.
  (c) If the max flow is equal to the sum of the team members, output TRUE, else output FALSE.

  (**+3** for this section / **+1** if it is not mentioned that the algorithms runs in polynomial time, i.e. the maximum flow is polynomial w.r.t input graph)
- Explanation of the running time: The running time of the algorithm depends on the creation of the graph and the running time of Ford Fulkerson's algorithm. Noticed the induced graph between teams and table can be a complete bipartite graph. So the edge complexity of the graph can be $O(mn)$ in the worst case. The running time of Ford Fulkerson's algorithm is $O(|E| \times f)$ where $|E|$ is the number of edges in the graph and f is the maximum flow. Now here, since the maximum flow here can be maximum $O(mn)$, the total complexity of the graph is $O(m^2n^2)$.

4. **(12 Marks)** Define the *decision versions* of the following optimization problems. Each parts **(a), (b), (c)** have 4 marks.

   (a) DOMINATING SET: Given an undirected graph $G = (V, E)$, find a set S with minimum number of vertices such that for every vertex $v \in V(G)$, either $v \in S$ or there is $u \in S$ such that $(u, v) \in E(G)$.

   Answer : The decision version of the problem is follows. Given an undirected graph G and an integer k, the problem is to find if there exists a set S of vertices with $|S| \leq k$ such that for every vertex $v \in V(G)$, either $v \in S$ or there is $u \in S$ such that $(u, v) \in E(G)$.

   Rubric : +4 for correct / **0** for incorrect.

   (b) CLIQUE: Given an undirected graph $G = (V, E)$, find a set S with maximum number of vertices such that for every $u, v \in S$, $(u, v) \in E(G)$.

   Answer : The decision version of the problem is follows. Given an undirected graph G and an integer k, the problem is to find if there exists a set S of vertices with $|S| \geq k$ such that for every $u, v \in S$; $(u, v) \in E(G)$.

   Rubric : +4 for correct / **0** for incorrect.

   (c) CLUSTER VERTEX DELETION SET: Given an undirected graph $G = (V, E)$, find a set S with minimum number of vertices such that every connected component of $G - S$ (i.e. after removing the vertices of S and the edges incident to S) is a clique (what it means is that if C is a connected component of $G - S$, then for every $u, v \in C$, $(u, v) \in E(G)$).

   Answer : The decision version of the problem if follows. Given an undirected graph G and an integer k, the problem is to find a set S of vertices with $|S| \leq k$ such that every connected component of $G - S$ is a clique.

   Rubric : +4 for correct / **0** for incorrect.