CS345A: Design and Analysis of Algorithms
End Semester Exam

Marks = 56 Date: 14 November 2023

NAME: Divyansh

ROLL No: 210355

34

Important Instructions:

1. You might lose 2 marks if you write answer to any question outside the box assigned for that question.

1. (Easiest problem) (*5 marks*)
Let $A[1..n]$ be an array storing n distinct positive numbers. We wish to determine if there exists any three distinct indices $1 \leq i, j, k \leq$ such that $A[i] + A[j] = A[k]$. Design an $O(n^2 \log n)$ time algorithm for this problem.

```
We sort the array A. Then.
for i in (~~1 to n~~) (1 to n-1)
    for j in (i+1 to n)
        binary search (A) for (A[i] + A[j])
        If found then return "Yes".
eventually if couldn't find then return No
```

5

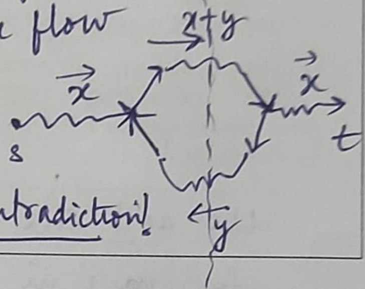
2. (Maximum flow) (*3,3 marks*)
G is a directed network with positive capacities on edges, and f is a valid flow from source s to sink t in G. In the following box, provide the complete description of the edge set of the residual network G_f.

Let E' be the edge set of G_f.
Now for $(x,y) \in E$ (the edge set of G)
If $f(x,y) < c(x,y)$,
$(x,y) \in E'$, with $c'(x,y) = c(x,y) - f(x,y)$
If $f(x,y) > 0$
$(y,x) \in E'$, with $c'(y,x) = f(x,y)$

3

Consider the following assertion:
There exists a maximum flow from s to t such that for every cycle C in G, at least one edge of C carries zero flow.
If the assertion is correct, produce a short justification for the assertion in the following box; otherwise produce a counterexample.

The assertion is correct. Let us assume that all the edges of the cycle is carrying non-zero flow. So, in this snapshot of graph, x flow is originating from ~~graph~~ source, but (x+y) is flow in upper cycle. Which is greater than originated. Contradiction!



0

3. (**Topological Ordering**) (*6 marks*)
G is a directed acyclic graph on n vertices and m edges given in the form of Adjacency Lists. You are also given an array InDegree[] that stores the number of incoming edges for each vertex. It is given that there is exactly one vertex s in G with InDegree = 0. The following is a pseudocode of the $O(m+n)$ time algorithm for computing topological ordering τ of G. Fill in the blanks carefully.

Algorithm 1 TolologicalOrdering(G, s)

```
$i \leftarrow 1$;
Enqueue($Q, s$);
while (...Q is not empty.........) do
    $x \leftarrow$ ....Dequeue(Q)..........;
    $\tau[x] \leftarrow$ ....i..................;
    $i \leftarrow i + 1$;
    foreach $(x, y) \in E$ do
        InDegree[$y$] $\leftarrow$ ..InDegree[y]-1.........;
        if (...InDegree[y] == 0..........) then
            .........Enqueue(Q, y).....
        end
    end
end
```

6

4. **Approximation Algorithms** (*2,3 marks*)

(a) In the following box, provide formal arguments to show that the cost of TSP tour in an undirected weighted complete graph G is greater than or equal to the cost of the MST of the graph.

Let us assume that TSP < MST and in TSP we return to the vertex from where the tour was started visited every other vertex once. Now remove the last edge from the tour (which connects to source). The remaining tour is a valid path spanning over all the vertices ⇒ we have MST with smaller cost. Contradiction! ✓

2

(b) Let $G = (V, E)$ be an undirected unweighted graph. Let E' be a maximal matching. Let S be the set of all endpoints of E'. In the following box, provide formal arguments to show that S is a vertex cover of G.

Let us focus on the vertices $V \setminus S$ (non endpoints). & let's call them X, $X \subseteq V$. Now each vertex of X had an edge with one of endpoints in E' (definition of maximal matching). which implies S either have an edge with vertex of G (the X ones) or is constituted by them (the S ones). Hence, S is a vertex cover of G.

✓ Or no edge incident on it

3

5. The following problems are on matrix multiplication and graph algorithms.

(a) **(On multiplying adjacency matrix)** *(2,2 marks)*
Let A be the adjacency matrix of an undirected graph on n vertices. It is also given that n is even. Let $C = A \times A$. Answer the following questions.

i. What parameter associated with vertex i is stored in $C[i, i]$? Write your answer in the following box.

degree of vertex i

2

ii. What is the minimum number of edges needed in G to ensure that there exists two integers i, j with $i \neq j$ such that $C[i, j] \neq 0$? Write exact (not asymptotic) answer in the following box.

$\frac{n}{2} + 1$

2

(b) **(Counting triangles efficiently)** *(8 marks)*
Let $G = (V, E)$ be an undirected graph on n vertices represented in the form of $n \times n$ Adjacency Matrix A. Three vertices i, j, k are said to form a triangle in G if $(i, j), (j, k), (i, k) \in E$. There is a blackbox B that receives any two $n \times n$ integer matrices and it outputs their integer product. You are allowed to make at most 1 call to this black box. Design an algorithm that outputs the count of all the triangles present in G. The time complexity of the algorithm, excluding the time taken by the black box, has to be $O(n^2)$.
Note: If your algorithm makes multiple calls to the black box, you will get at most 6 marks.

Algorithm: We Compute the A^2 matrix by calling the balackbox B. Now, Let C be an n-length array and I want the diagonal ~~ev~~ elements of $A3$ in C. So, C[i] is equal to dot product of ith row of A^2 and ith column of A. Once we have the C-array calculated.

$$\# \text{triangles} = \frac{(\text{sum of elements of C})}{6}.$$

5

6. **Learning from a Quiz** *(1,1,7,1,1,1 marks)*
Let $G = (V, E)$ be any undirected, unweighted, and connected graph on $n = |V|$ vertices and $m = |E|$ edges. Let $\delta(u, v)$ denote distance between u and v in G. Let M be a $n \times n$ matrix M such that

$$\delta(u,v) \le M[u,v] \le \delta(u,v) + 2 \quad \text{for each} \quad u, v \in V \tag{1}$$

2

M is said to store all-pairs distances with additive error 2. There is a fastest subcubic algorithm for computing M for G. The algorithm has 3 steps. The first 2 steps are given below. You need to design the 3rd step. Note that r is a parameter used in the algorithm. A vertex is said to be light if its degree is at most r; otherwise it is said to be heavy.

Step 1. We compute a dominating set $D \subset V$ of size $O((n \log n)/r)$ for the set of all heavy vertices, i.e., for each heavy vertex v, either $v \in D$ or at least one neighbor of v is present in D. Assume that D can be computed in $O(m \log n)$ time. For each vertex $v \in D$, we perform BFS traversal starting from v in G and store its exact distance to every other vertex in M. In the following box, state the time complexity of Step 1 in terms of m, n, r.

$\boxed{(m+n)n\log n / r}$ ✓

Step 2. For each pair of vertices i and j such that neither i nor j belong to D, perform the following task.

$$M[i,j] = \min_{r \in D}(M[i,r] + M[r,j])$$

In the following box, state the time complexity of Step 2 in terms of m, n, r.

$\boxed{\left(n - \frac{n\log n}{r}\right)^2 \frac{n\log n}{r}}$ ✓

Step 3. After this step, M will store all-pairs distances with additive error 2. You will have to compute a suitably *sparse* subgraph (V, E'), and suitably process *some* vertices in this subgraph. In the following box, describe the subgraph formally and the details of how you will process the vertices to update M.

Let's call this sparse graph G_2. Also $\bar{D}$ be the set of vertices v such that $v \in V$ and $v \notin D$. Now for every edge $(x, y) \in E$ if $x \in \bar{D}$ and $y \in \bar{D}$ then $(x, y) \in E'$. Now with the edge set E' ~~we perform~~ BFS starting from each vertex $v \in \bar{D}$ and update the corresponding distances in M.

In the following box, state the time complexity of Step 3 in terms of m, n, r.

$\left(m+n-\frac{n\log n}{r}\right)\left(n-\frac{n\log n}{r}\right)$

In the following box, find systematically the optimal value of r for which the time complexity of the entire algorithm is minimum.

total := $\left(m+n-\frac{n\log n}{r}\right)\left(n-\frac{n\log n}{r}\right) + \left(n-\frac{n\log n}{r}\right)^2 \frac{n\log n}{r}$

$+ (m+n)\frac{n\log n}{r}$

clearly $r = (\log n)$ minimizes this time complexity.

In the following box, state the time complexity of the algorithm for this optimal value of r.

$O(mn+n^2)$

7. **The power and limitations of algorithms**
Think wisely and choose that problem, out of the following 2 problems, which you feel fully confident of solving, and then attempt it. If you attempt both, you will get 0 marks.

4

(a) (*4 marks*) There is an infinite road along east-west direction. There is another infinite road along north-south direction. You are standing at the intersection of these two roads. Your friend is standing stationary on one of these infinite roads. But you know neither her distance from you nor the direction along which she is standing. You have a cycle which you may drive the way you want. Design an algorithm to reach your friend.

Step 3. After this step, M will store all-pairs distances with additive error 2. You will have to compute a suitably *sparse* subgraph (V, E'), and suitably process *some* vertices in this subgraph. In the following box, describe the subgraph formally and the details of how you will process the vertices to update M.

Let's call this sparse graph G_2. Also $\bar{D}$ be the set of vertices v such that $v \in V$ and $v \notin D$. Now for every edge $(x, y) \in E$ if $x \in \bar{D}$ and $y \in \bar{D}$ then $(x, y) \in E'$. Now with the edge set E' ~~we perform~~ BFS startingfrom each vertex $v \in \bar{D}$ and update the corresponding distances in M.

In the following box, state the time complexity of Step 3 in terms of m, n, r.

$\boxed{\left(m+n-\frac{n\log n}{r}\right)\left(n-\frac{n\log n}{r}\right)}$ ✗

In the following box, find systematically the optimal value of r for which the time complexity of the entire algorithm is minimum.

total := $\left(m+n-\frac{n\log n}{r}\right)\left(n-\frac{n\log n}{r}\right) + \left(n-\frac{n\log n}{r}\right)^2\frac{n\log n}{r}$

$+ (m+n)\frac{n\log n}{r}$

clearly $r = (\log n)$. minimizes this time complexity.

In the following box, state the time complexity of the algorithm for this optimal value of r.

$\boxed{O(mn+n^2)}$ ✗

7. **The power and limitations of algorithms**
Think wisely and choose that problem, out of the following 2 problems, which you feel fully confident of solving, and then attempt it. If you attempt both, you will get 0 marks.

4

(a) *(4 marks)* There is an infinite road along east-west direction. There is another infinite road along north-south direction. You are standing at the intersection of these two roads. Your friend is standing stationary on one of these infinite roads. But you know neither her distance from you nor the direction along which she is standing. You have a cycle which you may drive the way you want. Design an algorithm to reach your friend.

(b) (*10 marks*) The problem is the same as (*a*) except that your friend is not stationary. Moreover, he/she is walking along the road in a fixed direction at a constant speed which is c times the speed of your cycle for some real number $0 < c < 1$. The value of c is known to you. Which of the following statements is correct ? You must justify your answer.

i. There exists an algorithm to reach your friend.
ii. There is a constant $c_0 < 1$ such that if $c < c_0$, then there exists an algorithm to reach your friend; otherwise there does not exist any algorithm to reach your friend.
iii. If your friend is walking away from the point of intersection of the roads, there does not exist any algorithm to reach your friend.

In the following box, state the problem (a or b) you are attempting.

(a)

In the following box, provide the solution of the problem you are attempting.

I will start cycling in any of the 4 directions. First I go 1 unit distance then I come back if not found. Then I do this for other 3 directions. Now again, I go 2 unit and comeback if not found and again repeat this in the other 3 roads. ~~I~~ Again I go 4 units, then 8 units and so on untill I find my friend in any one of the road.

Advice: It might be a bad idea to be greedy for marks while solving Problem 7. Act wisely and attempt only that part (a or b) for which you are fully confident.