1 9

CS345A: Design and Analysis of Algorithms

Quiz 5

Marks = 20

Date: 7th November 2023

NAME: Divyansh

ROLL No: 210355

Important Instruction:
You might lose 1 mark if you write answer to any question outside the box assigned for that question.

Question 1 Attempt any one of the following problems.

1. Easy Problem (*5 marks*)
   There is a directed graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges, where each vertex has a label which is a real number. For each vertex v, you wish to compute the the label of the maximum label vertex reachable from it. Design an $O(m + n \log n)$ time algorithm which outputs n pairs $\{(v, L(v))|v \in V\}$, where $L(v)$ denotes the maximum label vertex reachable from v.

Given the directed graph, we generate the DAG graph G_C consisting of strongly connected components as the new set of vertices.
Now, within a SCC calculate the vertex with highest label and for all vertices v in the SCC assign $L(v)$ as the highest label of that SCC.

5

Now, the get the topological order of G_c (the SCC graph) and start iterating from right end. If the SCC node to the left of the curr SCC has an edge to it then update the highest label of that SCC if it is originally ~~origina~~ smaller.
The idea here is that if there is an edge from SCC S_1 to SCC S_2 then all the vertices of S_2 are reachable from all the vertices of S_1 and hence $Label(S_1) \leftarrow max(Label(S_1), Label(S_2))$.

2. Hard Problem *(10 marks)*

Let $G = (V, E)$ be an undirected connected graph on $n = |V|$ vertices and $m = |E|$ edges. Each edge has a capacity which is a nonnegative real number. Let $A \subset V$ such that $\emptyset \neq A \neq V$. A defines a cut, denoted by $\text{cut}(A, \bar{A})$, which consists of all edges with exactly one endpoint in A. Capacity of $\text{cut}(A, \bar{A})$ is the sum of the capacities of all the edges that belong to it. A cut $(A, \bar{A})$ is said to be a cut for a pair of vertices u and v if $u \in A$ and $v \notin A$ or vice versa. A cut $(A, \bar{A})$ for u and v is said to be a minimum cut for u and v if its capacity is less than or equal to the capacity of every other cut for u and v. Let $\mu(u, v)$ denote the capacity of the minimum cut for u and v.

Consider the set $M = \{\mu(u, v) | u, v \in V\}$. Although there are $\binom{n}{2}$ possible pairs of vertices, cardinality of M even in the worst case is much smaller than $\binom{n}{2}$. Your aim is to establish the tightest bound on worst case size of M.

Hint: Choose a pair of vertices suitably and then proceed using some well-known algorithm paradigm (also studied in the class). Exploit the undirectedness of the graph.

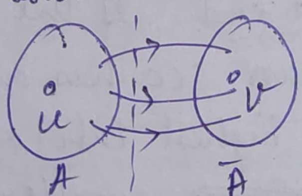
The bound on $\|M\|$:
$n-1$

Provide Justification for your answer in the following box. (marks will be deducted heavily for vague, incomplete, or informal arguments).

0

~~Selecting a~~

Ideation: the minimum cut $(A, \bar{A})$ for any pair of vertices (u, v) shall be obtained from maximum flow with u as the source and v as the destination. Now,



Let us consider this instances, all the vertices in A is reachable from the source and hence the $\mu(u', v')$ for any $u' \in A$ and $v' \in \bar{A}$ will be same as $\mu(u, v)$. Now the next combination of pairs will be obtained by partioning A further as we did with the whole graph. For maximum bound case we can $n/2$ cuts at last left.

So $|M| \leq \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots \leq n-1$

4

Question 2

Attempt any one of the following 2 questions.

1. Easy Problem (*2.5,2.5 marks*)

(a) While discussing the topic of amortized analysis, we introduced the concept of potential function. We defined the amortized cost of an operation as the sum of the actual cost of the operation and the change in the potential function during that operation. In the following box, state the properties that must be satisfied by a potential function, and use them suitably to establish that the amortized cost of any sequence of m operations is an upper bound on the actual cost of the sequence of m operations.

For any potential function ϕ, the value of $\phi(0) = 0$ which is the initial potential, And the potential after ~~for~~ the further operations should be non-negative.
Let there be set ~~of~~ O of m operations, $O = \{O_1, O_2, \ldots O_m\}$
Let $T(i)$ be the actual cost of ith operation
and $A(i)$ be the amortized cost of ith operation.
from definition of Ammortized cost,

$$A_i = T_i + \phi_i - \phi_{i-1}$$

suming the above equation for m operations,

$$\Sigma A_i = \Sigma T_i + \phi_m - \phi_0 \quad \text{(rest } \phi \text{ got cancelled)}$$

we have $\phi_0 = 0$ and $\phi_m \geq 0$ so, $\boxed{\Sigma A_i \geq \Sigma T_i}$

(b) Recall the problem of counting the number of bit-flips of a binary counter during n increment operations discussed in the class. The binary counter is initialized with value 0. Consider the following potential function at the end of i increment operations.

$\phi(i)$: the length of the longest suffix of all 1's in the binary representation of i

Does it provide an $O(1)$ bound on the amortized number of bit-flips for the increment operation ? Provide your answer (yes or no) along with proper justification in the following box.

Yes, ~~this~~ potential provide a $O(1)$ bound on the amortized number of bit-flips for the increment.

In the incrementation, we have the condition where all the bits are 1, ~~here~~ these are the cases with high cost so, let's see the ith increment

Case	Actual cost	$\Delta\phi$	Amortized
Normal	c	c	$2c$
high cost	$(1+\log i)c$	$-\log i c$	c

so Ammortized $\leq 2c$ which is $O(1)$.

2. (*marks=4,2,2,2*)

Let G be a directed graph on n vertices, numbered from 1 to n, with no edges initially. Let s be a designated source vertex. We receive an online sequence of m edge insertions, where $m > n$. Our aim is to maintain a Boolean array $R[1..n]$ with the following property for each $j \leq n$ after each edge insertion.

$R[j]$ = true if and only if there is a path from vertex s to vertex j consisting of edges present in the graph.

In the beginning $R[s]$ =true, and $R[i]$ =false for each $i \neq s$. Upon insertion of an edge (i, j), we invoke Procedure Update-R(i, j) to update R. This is an efficient recursive procedure sketched below. Fill in the blanks appropriately.

Procedure Update-R(i, j)

```
if (R[i] == true and R[j] == False) then
    R[j] ← true;                          — O(1)
    foreach neighbor q of j do            } #Neighb(j)
        Update-R(j,q)
    end
end
```

4

Though the worst case time complexity of updating R using Update-R() procedure after an edge insertion may be quite large, the total time complexity for processing m edge insertions is $O(m)$. You have to establish this fact using amortized analysis as follows.

State the potential function you use in the following box.

$\phi = |n - \#(\text{False value in } R) - 1|$ ✗

Express the actual time complexity of updating R using Update-R() procedure after an edge insertion formally and precisely in the following box.

1 + #(vertices reachable from j and with R value False) ✗

In the following box, show using the potential function defined above that the amortized cost of updating R using Update-R() procedure after an edge insertion is indeed $O(1)$.