CS 170 Homework 4

Due **2020-02-17, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

2 True Source

Design an efficient algorithm that given a directed graph G determines whether there is a vertex v from which every other vertex can be reached. (Hint: first solve this for directed acyclic graphs. Note that running DFS from every single vertex is not efficient.)

Please give a 3-part solution to this problem.

3 Finding Clusters

We are given a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$, i.e. the vertices are integers in the range 1 to n. For every vertex i we would like to compute the value $m(i)$ defined as follows: $m(i)$ is the smallest j such that vertex j is reachable from vertex i. (As a convention, we assume that i is reachable from i.) Show that the values $m(1), \ldots, m(n)$ can be computed in $O(|V| + |E|)$ time.

Please give a 3-part solution to this problem.

4 All Roads Lead to Rome

You are the chief trade minister under Emperor Caesar Augustus with the job of directing trade in the ancient world. The Emperor has proclaimed that *all roads lead to (and from) Rome*; that is, all trade must go through Rome. In particular, you are given a strongly connected directed graph $G = (V, E)$ with positive edge weights, and there is a particular node $v_0 \in V$ (Rome).

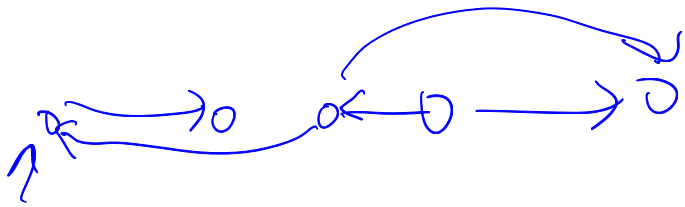
(a) Give an efficient algorithm for finding shortest paths between *all pairs of nodes*, with the one restriction that these paths must all pass through v_0 (Rome). Make your algorithm as efficient as you can (perhaps as fast as Dijkstra's algorithm).

Please give a 3-part solution.

(b) Occasionally, Augustus will ask you for the (smallest) distance between two vertices. You want to do this as quickly as possible, so that Augustus does not have your head.

This is called a *distance query*: Given a pair of vertices (u, v), give the distance of the shortest path from u to v that passes through v_0. Describe how you might store the

2. topogical sort, select the left most vertex and test it.



O(V + E).

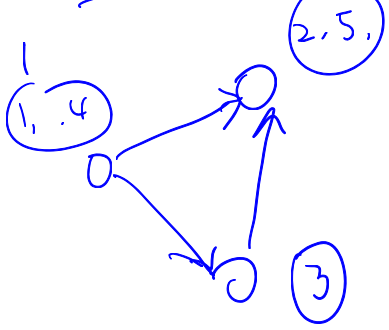
what about cycle? . test every vetex had a edge to it.

SCCs SCCs

is reachable to others.

meta graph

3. SCCs 2 1, 2, 3, 4, 5



4. (a) u, v. shortest path to v_0

origin graph from reverse Graph to

u shortest Path to v_0.

v Shortest Path to v_0.

use dijestra to find the shortest path tree is enough

(b) dist to v_0 [v] → update at dijestra . from and to is different

dist to v_0 [u] + dist to v_0 [v]

(c) edge to v_0 [v]

$u \leftarrow u_1 \leftarrow u_2 \in v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v$.

results such that you require $O(|V|)$ storage, and you can compute the result in $O(1)$ time. For your answer, a clear description of the data structure and its usage is sufficient.

(c) On the other hand, the traders need to know the paths themselves.

This is called a *path query*: Given a pair of vertices (u, v), give the shortest path from u to v that passes through v_0. Describe how you might store the results such that you require $O(|V|)$ storage, and you can compute the result in $O(|V|)$ time. Again, a clear description of the data structure and its usage is sufficient.

5 The Greatest Roads in America

Arguably, one of the best things to do in America is to take a great American road trip. And in America there are some amazing roads to drive on (think Pacific Crest Highway, Route 66 etc). An intrepid traveler has chosen to set course across America in search of some amazing driving. What is the length of the shortest path that hits at least k of these amazing roads?

Assume that the roads in America can be expressed as a directed weighted graph $G = (V, E, d)$, and that our traveler wishes to drive across at least k roads from the subset $R \subseteq E$ of "amazing" roads. Furthermore, assume that the traveler starts and ends at her home $h \in V$. You may also assume that the traveler is fine with repeating roads from R, i.e. the k roads chosen from R need not be unique.

Provide a 3-part solution with runtime in terms of $n = |V|$, $m = |E|$, k, and $r = |R|$.

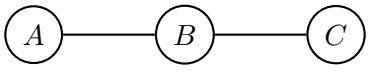
Hint: First consider $k = 1$. How can G be modified so that we can use a "common" algorithm to solve the problem?

6 Graph Game

Given an undirected, unweighted graph G, with each node v having a value $\ell(v) \geq 0$, consider the following game.

1. All nodes are initially *unmarked* and your score is 0.
2. Choose an unmarked node u. Let $M(u)$ be the *marked* neighbours of u. Add $\sum_{v \in M(u)} \ell(v)$ to your score. Then mark u.
3. Repeat the last step for as many turns as you like, or until all the nodes are marked.

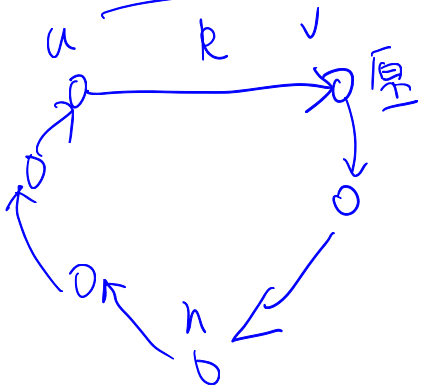
For instance, suppose we had the graph:



with $\ell(A) = 3$, $\ell(B) = 2$, $\ell(C) = 3$. Then, an optimal strategy is to mark A then C then B giving you a score of $0 + 0 + 6$. We can check that no other order will give us a better score.

5. change the graph.

Dijkstra (v→h). k=1 k=k.



原 Dijkstra (h)

shortest (v, →v)

u → v

Dijkstra (v, →h).

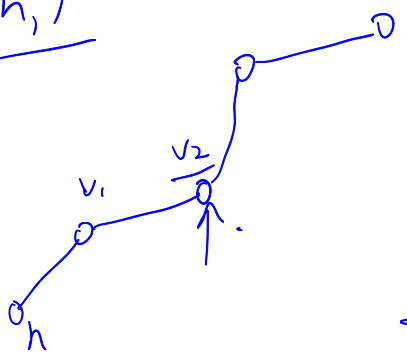
gredy ?

Dijkstra (h,)

shortest v.

k>2

to h?



k((E'+|V|) log|V|

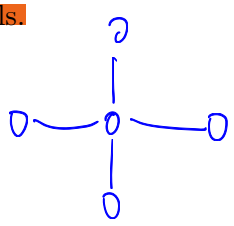
(a) Is the following statement true or false? **There is always an optimal strategy that marks all the nodes.** Briefly justify your answer.

(b) Give a greedy algorithm to find the order which gives the best score. Describe your algorithm, prove its optimality, and analyse its running time.

(c) Now suppose that $\ell(v)$ can be negative. Give an example where your algorithm fails.

(d) Your friend suggests the following modified algorithm: delete all v with $\ell(v) < 0$, then run your greedy algorithm on the resulting graph. Give an example where this algorithm fails.

(a)

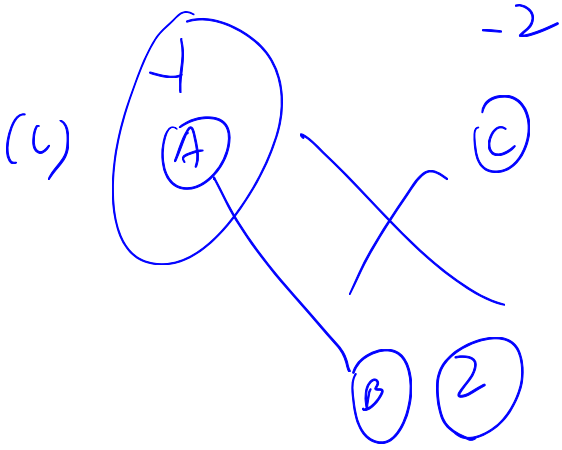


?

(b)

First mark the biggest $\ell(v)$?

(c)



1