

# CSE548/AMS542 Fall 2008 Analysis of Algorithms

Jie Gao\*

October 2, 2008

Due **September 18th** before class. Each problem, unless specified otherwise, has a maximum of 10 points. (i) You write down the solution clearly. If we can not recognize your writing then you may lose points. (ii) Avoid too many details. A succinct and clean proof is the best. You may use the algorithms we covered in class without referring to the details. (iii) If you discuss some of the problems with another fellow student (at most 2 students per group), write down his/her name and the problems. If you consult any books/webpages, cite them.

## Homework 1

1. Prove or disprove (i.e., give counter examples) for the following claims.  $f(n), g(n)$  are non-negative functions.

(a)  $\max(f(n), g(n)) = \Theta(f(n) + g(n))$ .

**true**  $\frac{1}{2}(f(n)+g(n)) \leq \max(f(n),g(n)) \leq f(n)+g(n)$

(b)  $o(f(n)) \cap \omega(f(n)) = \emptyset$ .

**true**  $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = 0$  and  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

(c)  $(n+a)^b = \Theta(n^b)$ ,  $a, b$  are positive integers.

**true**  $\lim_{n \rightarrow \infty} \frac{(n+a)^b}{n^b} = 1$

(d)  $f(n) = O(f(n)^2)$ .

**false**  $n^{-1} = \omega(n^{-2})$

(e)  $f(n) = O(g(n))$  implies that  $2^{f(n)} = O(2^{g(n)})$ .

**false**  $f(n) = n, g(n) = n/2$ .  $f(n) = O(g(n))$ , but  $2^{f(n)} = 2^n = \omega(\sqrt{2}^n) = 2^{g(n)}$

2. Prove claim (3.2) on page 78.

Let  $G$  be an undirected graph on  $n$  nodes. Prove that any two of the following statements implies the third.

(a)  $G$  is connected;

(b)  $G$  does not contain a cycle;

(c)  $G$  has  $n - 1$  edges.

**(a)(b) $\implies$ (c):** By (a) and (b),  $G$  is a tree. A tree has  $n-1$  edges.

**(b)(c) $\implies$ (a):** Prove by contradiction. If  $G$  is not connected, then it is composed of  $k$  connected components ( $k \geq 2$ ). Divide  $n-1$  edges into these  $k$  components, since  $n - 1 > n - k$ ,

---

\*Department of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA, [jgao@cs.sunysb.edu](mailto:jgao@cs.sunysb.edu).

by pigeonhole principle, there will exist at least one component  $G_i$  with  $EdgeNumber(G_i) \geq VertexNumber(G_i)$ , which means  $G$  contains a cycle. Contradiction with (b).

**(c)(a) $\implies$ (b):** Prove by contradiction. Suppose  $G$  contains at least one cycle, then if we find this cycle and remove an arbitrary edge,  $G$  should still be connected. After removal,  $G$  has  $n-2$  edges. But if we start a graph  $G'$  with  $n$  vertices and no edge, there are  $n$  disconnected components. Each time we add one edge into  $G'$ , we at most reduce the disconnected components by one. So  $n-2$  edges cannot make a graph connected. Contradiction with (a).

3. The diameter of an unweighted graph  $G = (V, E)$  is the largest of the all shortest-path distances in the graph.

- (a) Give an algorithm to calculate the diameter of a graph. Analyze its running time.
- (b) Give an  $O(n + m)$  algorithm to *estimate* the diameter of a graph up to an approximation factor of 2. That is, calculate an approximate diameter  $d$  such that the true diameter is at least  $d$  and at most  $2d$ .

**Solution:**

- (a) Start at each vertex, do BFS or DFS, then take the max value of the  $n$  depth.  $O(n^2)$
- (b) Start at an arbitrary vertex  $u$ , do BFS or DFS, take the depth  $d$ . Proof: By definition of diameter,  $diameter \geq d$ ; for each pair of nodes  $v$  and  $w$ ,

$$shortest\_path(v, w) \leq shortest\_path(v, u) + shortest\_path(u, w) \leq 2d,$$

which means  $diameter \leq 2d$ .

4. A directed graph is semi-connected if for all pairs of vertices  $u, v$ , we have either a path from  $u$  to  $v$  or a path from  $v$  to  $u$ . Give an efficient algorithm to determine whether or not  $G$  is semi-connected. Also analyze the algorithm's running time.

**Method1:**

Step1: Reverse all edges in  $G$ , we get graph  $G'$ .

Step2: For each node  $v \in V(G)$ , do BFS or DFS on both  $G$  and  $G'$ , we get two sets of points  $Reachable_v(G)$  and  $Reachable_v(G')$ .

Step3: Check if  $Reachable_v(G) \cup Reachable_v(G') = V(G) - v$ . if false, then  $G$  is not a semi-connected graph, return false.

Step4: Repeat step 2 and 3 until all nodes are tested. Return true.

Reversing takes  $O(m+n)$ , BFS or DFS takes  $O(m+n)$ , Checking can be done in  $O(n)$ , and we have  $n$  nodes, so in total worst case running time is  $O(n(m+n))$ .

**Method2:**

Step1: Find all strongly connected components in  $G$ .

Step2: Shrink each strongly connected component into one node, we get a new graph  $G'$ , which can be easily shown to be a directed acyclic graph(DAG).

Step3: Do topological sorting on  $G'$ , get the sorted list of nodes  $v_{i_1}, v_{i_2}, \dots, v_{i_k}$ .

Step4: for each pair of consecutive nodes  $v_{i_j}$  and  $v_{i_{j+1}}$ , check if there exist an edge from  $v_{i_j}$  and  $v_{i_{j+1}}$ . If false,  $G$  is not a semi-connected graph, return false.

Step5: repeat step3 and 4 until all pair of consecutive nodes are tested. Return true.

**Brief proof of step4 and 5:**

( $\implies$ ) : If all pair of consecutive nodes have an edge going through them, then  $G'$  is semi-connected. Each node in  $G'$  is either one node in  $G$ , or a strongly connected component in  $G$ , so  $G'$  is semi-connected.

( $\Leftarrow$ ) : Suppose there exists  $j$  such that no edge going from  $v_{i_j}$  to  $v_{i_{j+1}}$ . Because  $G'$  is a DAG and  $v_{i_{j+1}}$  is after  $v_{i_j}$  in topological sorting, there is no path going from  $v_{i_{j+1}}$  to  $v_{i_j}$ , otherwise  $v_{i_{j+1}}$  should be a prerequisite of  $v_{i_j}$ . If there is a path going from  $v_{i_j}$  to  $v_{i_{j+1}}$ , let's say  $v_{i_j}, v_k, \dots, v_{i_{j+1}}$ , then  $v_k$  is a prerequisite of  $v_{i_{j+1}}$ , which means in topological ordering  $v_k$  should be between  $v_{i_j}$  and  $v_{i_{j+1}}$ , contradiction here. To sum up, if there is no edge going from  $v_{i_j}$  to  $v_{i_{j+1}}$ , then there is no path going between these two nodes.

**Running time:** all steps are linear, so running time is  $O(m+n)$ .

5. How can the number of strongly connected components of a graph change when an edge is added between two existing vertices. Give examples for each possible case and prove those are the only possible cases.

Two possibilities: No change or decrease by  $k$ .