

Homework 20190929

Due Date: 20191010, 10 P.M.

Ex2-1

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures.

a. $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

c. $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$, where $\lg(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n .

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

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

f. $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

g. $f(n) = \Theta(f(n/2))$.

Ex2-2

Show that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \lg n)$.

Ex2-3

Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m + n)$ for a graph with n nodes and m edges.

Ex2-4

We have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u , and obtain a tree T that includes all nodes of G . Suppose we then compute a breadth-first search tree rooted at u , and obtain the same tree T . Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u , then G cannot contain any edges that do not belong to T .)

Ex2-5

There's a natural intuition that two nodes that are far apart in a communication network—separated by many hops—have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.

Suppose that an n -node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. Show that there must exist some node v , not equal to either s or t , such that deleting v from G destroys all s - t paths. (In other words, the graph obtained from G by deleting v contains no path from s to t .) Give an algorithm with running time $O(m + n)$ to find such a node v .