

CMSC 451: Homework 3, Spring 2004

Due at the beginning of class on March 9; *note that this homework occupies two pages and has six problems in total.*

Warning: some of the problems require thought - do not wait until the last day to start working on them!

If you cannot come up with algorithms that run in the required time, then provide (correct) slower algorithms for partial credit. Write your answers using *pseudo-code* in the same style as the textbook. These make the algorithm description precise, and easy to read (as opposed to code in C or some other language).

1. You are given: (i) an *undirected* graph $G = (V, E)$ with a non-negative weight for each edge; (ii) a vertex $r \in V$, and (iii) a non-negative real number h .
 - (a) Design as efficient an algorithm as you can, to construct a spanning tree T of G with the root of T being r , for which the following holds: the weight of the path in T from r to *every* vertex u , is at most h . (Note that we are not talking about an arbitrary path between r and u , but the path in T between r and u .) If no such tree T exists, your algorithm should say so.
 - (b) Analyze the time complexity of your algorithm.
2. Do problem 33.4-3 from page 962 of the textbook.
3. We are given an array A of n integers where n is at least 2, and is a power of two; all elements of A are distinct. We want to find the pair (min, max) , where min is the smallest element in A and max is the largest element in A . Consider the following divide-and-conquer algorithm for this problem.

if $(n = 2)$ then:

if $(A[1] < A[2])$ then return $(A[1], A[2])$;
else return $(A[2], A[1])$;

else if $(n > 2)$ then:

solve the problem recursively on the first half of A , and let $(min1, max1)$ be the output obtained by this recursive call;
solve the problem recursively on the second half of A , and let $(min2, max2)$ be the output obtained by this recursive call;
if $(min1 < min2)$ then:

if $(max1 > max2)$ then return $(min1, max1)$;
else return $(min1, max2)$;

else

if $(max1 > max2)$ then return $(min2, max1)$;
else return $(min2, max2)$.

Write a recurrence relation for the number of comparisons $C(n)$ made by this algorithm, and prove that $C(n) = 3n/2 - 2$. ($C(n)$ includes all the comparisons, **except** the comparisons “if ($n = 2$)” and “else if ($n > 2$)”.)

4. Do problem 28.2-5 from page 741 of the textbook.
5. Do problem 28.2-6 from page 741 of the textbook.
6. **(Only for graduate students)** We are given a directed graph $G = (V, E)$ with a *non-negative* weight $w(u, v)$ given for each edge (u, v) ; a vertex $s \in V$ is also specified. Assume that there is a path from s to every other vertex. Suppose we run Dijkstra’s algorithm for single-source shortest paths from s to all other vertices. Suppose u_1, u_2, \dots, u_n is the order in which the vertices are added to the set S over time, where n is the number of vertices in G . Is it true that once the algorithm has finished, we have $d[u_1] \leq d[u_2] \leq \dots \leq d[u_n]$? Justify your answer.