OLD CS 473: Fundamental Algorithms, Spring 2015

Discussion 5

February 19, 2015

5.1. Separators in trees.
Divide and conquer is a basic algorithmic strategy. To apply it to graphs, one need some additional properties about how the pieces of the graph interact. This is not always possible for all types of graphs, but trees are simpler. To this end, consider a tree $T = (V, E)$ on n nodes. A node v is a balanced ***separator*** if removing v from T results in a forest in which no tree of the forest has more than $2n/3$ nodes. Prove that every tree T has a balanced separator.
Describe a linear time algorithm to find such a separator. This can be generalized to the case where nodes have weights and the balance is with respect to the weights on the nodes. Can you formulate such a theorem?

Hint: Root T and consider the root and if it does not work pick an appropriate child etc.

Edge separator. Some times it is preferable to remove an edge of the tree. Specifically, an edge e of T is a balanced *separator* if removing e results in two trees with each having at most $2n/3$ nodes. Give an example of a tree without such a separator. Suppose all nodes in T have degree at most $d+1$. Show that there is an edge e such that $T-e$ has no component with more than $(1-1/d)n$ nodes.

5.2. Using separators in trees.
(Extra problem - not for the discussion itself.)
Given a tree T with weights on the edges, describe how to build a data-structure with $O(n \log n)$ space, such that given any two vertices $u, v \in \mathsf{V}(T)$, the data-structure computes in $O(\log n)$ time, the length of the shortest path in T between u and v.

5.3. Largest Complete Subtree.
A ***subtree*** of a binary tree is a connected subgraph, see Figure 5.1. A binary tree is ***complete*** if every internal node has two children, and every leaf has exactly the same depth. Describe and analyze a recursive algorithm to compute the *largest complete subtree* of a given binary tree. Your algorithm should return the root and the depth of this subtree.

5.4. Rod Cutting.
Suppose we are given a steel rod of length n. Also, assume we are given an array $p[1 \ldots n]$, where $p[i]$ is the price a rod of length i sells for. Given that we can make cuts for free (and that we only cut integer lengths), provide an algorithm that efficiently computes the maximum amount we can make by cutting up and selling our rod of length n.

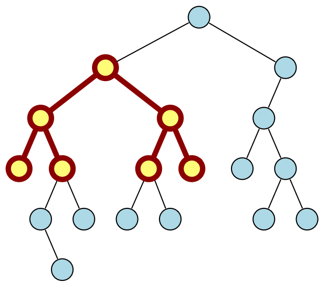

Figure 5.1: The largest complete subtree of this binary tree has depth 2.

5.5. Billboards.

Consider a stretch of Interstate-57 that is m miles long. We are given an increasing sorted list of mile markers, $x_1 \le x_2 \le \cdots \le x_k$ in the range 0 to m, at each of which we are allowed to construct billboards (suppose they are given as an array $X[1 \ldots k]$). Suppose we can construct billboards for free, and that we are given an array $R[1 \ldots k]$, where $R[i]$ is the revenue we would receive by constructing a billboard at location $X[i]$. Given that state law requires billboards to be at least 5 miles apart, give an efficient algorithm to compute the maximum revenue we can acquire by constructing billboards. Solve this by dynamic programming. To help arrive at recurrence consider solving the problem by divide and conquer. Think of why the two subproblems depend on each other and how to encapsulate this dependence.

5.6. k partition.

Given a set of $S = \{\alpha_1, \ldots, \alpha_n\}$ of n positive integers in the range 1 to m, decide, given a parameter k, if the numbers in S can be partitioned into k sets $S_1, \ldots, S_k$, such that, for all i, $w(S_i) = w(S)/k$, where $w(X) = \sum_{x \in X} w(x)$. What is the running time of your algorithm as a function of n, m and k.