CS140: Algorithms

Fall 2000

Assignment 9

This assignment is due by 5 PM on Friday 10/27. You may collaborate with your classmates to complete this assignment but you must reference your collaborators. In addition, you must do your own write-up. This assignment is worth 100 points (50 points if handwritten). You may, however, submit a handwritten solution to the network flow problem.

Your answers will be judged for correctness, clarity, and conciseness. When you give a recursive definition or algorithm, you should begin with a brief (a few sentences) overview that identifies the key ideas behind your approach. Then fill in the (important) details.

1. In your humsoc class you are assigned 7 books to read in the next four days. The books build on one another so you must finish the i-th book before you begin the $i+1$-st book. You have this novel idea that you could *spread* the workload across the week instead of doing it all on the very last day! Suppose you decide to finish any book you start within the same day it (you have trouble picking up where you left off) and that the books have $300, 200, 100, 900, 20, 890$, and 55 pages respectively. How should you split up the reading between the four days to minimize the maximum number of pages you have to read per day?

   This works out so well you decide to sell your work scheduling services to your fellow classmates. But you need a scheduling algorithm for the problem with n books containing, respectively, $p_1, p_2, \ldots, p_n$ pages to be read in k days. As before the books must be read in order and each book must be finished in the same day it is started. You realize there are exponentially many possible schedules and (given the HMC workload) you don't have time to look at them all.

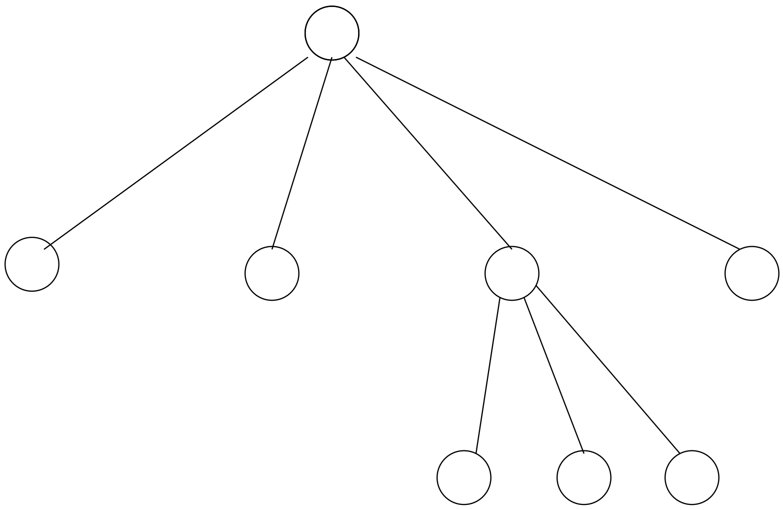
   Let $T(i, j)$ be the minimized maximum reading per day to read books 1 through i in j days.

   (a) Give a recursive definition for $T(i, j)$ including any necessary base cases.

   (b) Give a dynamic programming algorithm for computing $T(n, k)$.

   (c) Analyze the running time of your algorithm.

2. A dominating set in a graph $G = (V, E)$ is a subset of vertices $W \subseteq V$ such that every vertex v in G is either in W or connected by an edge to a vertex in W. The problem of finding a smallest dominating set in a

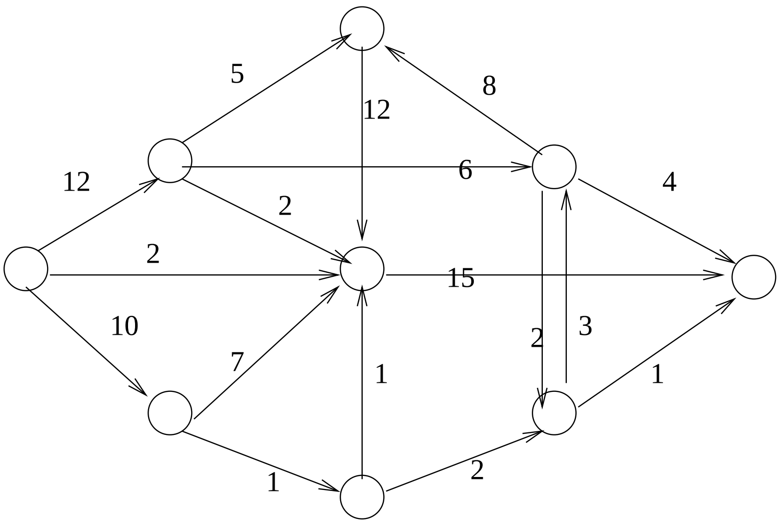


graph is NP-hard. The problem is tractable, however, when the input is restricted to be a tree.

Find the smallest dominating set of the tree shown in the figure above.

Let T be a tree. Choose an arbitrary node r in T and root T at r. For each node $v \in T$ let $D(v)$ denote the size of the smallest dominating set of the subtree rooted at v. Give a recursive definition for $D(v)$. Hint 1: Strengthen the inductive hypothesis; i.e. you'll need to define (recursively) some additional functions at the nodes of T. Hint 2: Visualize a solution W and consider the vertices in W from the subtree rooted at v. Is this set necessarily a dominating set of the subtree? Under what conditions does it include/exclude v?

3. Find a maximum flow in the network shown on the next page by using the augmenting path method. At each stage show the network with current flows and the residual graph.


5
8
12
12
6
4
2
2
15
10
3
2
7
1
1
1
2