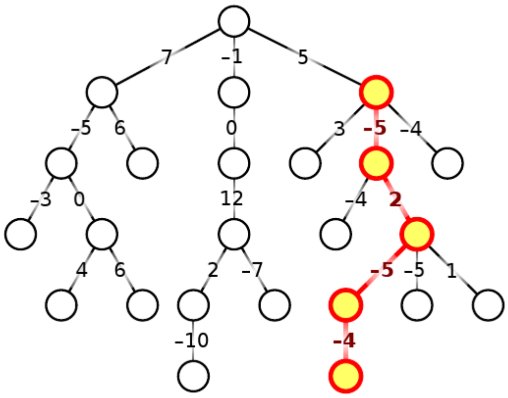
CS 473: Algorithms, Fall 2009
HBS 8

1. **[DP in a tree]**
Let T be a rooted tree with integer weights on its edges, which could be positive, negative, or zero. For the unique path between nodes x, y, we define its length, $l(x, y)$, to be the sum of the weights of its edges. Finally, let $D(x)$ denote the set of descendents of node x. Design a linear time algorithm to find the minimum, over all nodes $x \in T$, of the minimum-length path from x down to one of its descendants. That is, compute $\min_{x \in T} \min_{y \in D(x)} l(x, y)$. For example, given the tree shown below, your algorithm should return the number -12.



The minimum-weight downward path in this tree has weight -12.

2. **[Pseudo-Polynomial time for** PARTITION**]**
Consider the following PARTITION problem. Given integers $a_1, \ldots, a_n$, we want to determine whether it is possible to partition $\{1, \ldots, n\}$ into two disjoint subsets I, J such that

$$\sum_{i \in I} a_i = \sum_{j \in J} a_j = \frac{1}{2} \sum_{i=1}^{n} a_i$$

For example, for input $(1, 2, 3, 4)$ the answer is *yes*, because there is the partition $(1, 4), (2, 3)$. On the other hand, for input $(2, 2, 3, 4)$ the answer is *no*. Devise and analyze a dynamic programming algorithm for PARTITION that runs in time polynomial in n and in $\sum_i a_i$.

3. **[Making Change.]**
Suppose you are a simple shopkeeper living in a country with n different types of coins, with values $1 = c[1] < c[2] < \cdots < c[n]$. (In the U.S., for example, $n = 6$ and the values are 1, 5, 10, 25, 50, and 100 cents.) Your beloved benevolent dictator, El Generalissimo, has decreed that whenever you give a customer change, you must use the smallest possible number of coins, so as not to wear out the image of El Generalissimo lovingly engraved on each coin by servants of the Royal Treasury.

(a) Give a dynamic programming algorithm to determine, given a target amount A and a sorted array $c[1..n]$ of coin values, the smallest number of coins needed to make A cents in change. You can assume that $c[1] = 1$, so that it is possible to make change for any amount A.

(b) **Note**: We don't expect you to complete the rest of this problem in HBS. For more practice, you can think about them at home.

Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

(c) Suppose that the available coins have the values $v^0, v^1, \ldots, v^k$ for some integers $v > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

(d) Give a set of 4 coin values for which the greedy algorithm does not yield an optimal solution, show why.