CS170 Discussion Section 7: *3/8*

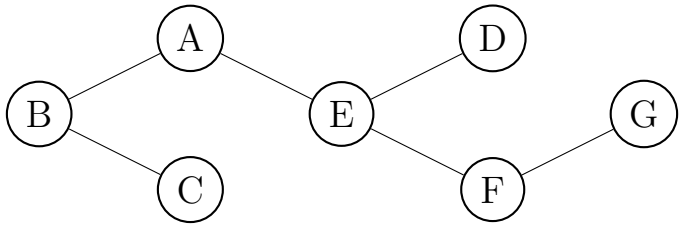
Vertex cover

A *vertex cover* of a graph $G = (V, E)$ is a subset of vertices $S \subseteq V$ that includes at least one endpoint of every edge in E. Give a linear-time algorithm for the following task:

Input: An undirected *tree* $T = (V, E)$.

Output: The size of the smallest vertex cover of T.

For instance, in the following tree, possible vertex covers include $\{A, B, C, D, E, F, G\}$ and $\{A, C, D, F\}$ but not $\{C, E, F\}$. The smallest vertex cover has size 3: $\{B, E, G\}$



Pig

Pig is a 2-player game played with a 6-sided die. On your turn, you can decide either to roll the die or to pass. If you roll the die and get a 1, your turn immediately ends and you get 1 point. If you instead get some other number, it gets added to a running total and your turn continues (i.e. you can again decide whether to roll or pass). If you pass, then you get either 1 point or the running total number of points, whichever is larger, and it becomes your opponent's turn. For example, if you roll 3, 4, 1 you get only 1 point, but if you roll 3, 4, 2 and then decide to pass you get 9 points. The first player to get to 100 points wins.

Suppose at some point the player whose turn it is has x points, their opponent has y points, and the running total for this turn so far is z. Let's work out what the optimal strategy is.

1. Let $W(x, y, z)$ be the probability that the current player will eventually win if both players play optimally. What is $W(x, y, z)$ if $x + z \geq 100$?

2. Suppose the current player decides to roll, and gets a 1. What's the probability that they'll still win, in terms of the function W?

3. Give a recursive formula for $W(x, y, z)$.

   Hint: Work out the probabilities R and P that the current player will win if they decide to roll and pass respectively. Their optimal move is to do whichever of these would give the greatest winning probability, so W will be the maximum of R and P.

4. Describe a dynamic programming algorithm to compute $W(x, y, z)$. If you needed N points to win instead of 100, what would be the asymptotic runtime of your algorithm?

Longest common subsequence

Given two strings $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_m$, we wish to find out the length of their *longest common subsequence*, that is, the largest k for which there are indices $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ with $x_{i_1}x_{i_2} \dots x_{i_k} = y_{j_1}y_{j_2} \dots y_{j_k}$. For example, the longest common subsequence of "exponential" and "polynomial" is "ponial" with length 6. Show how to do this in time $O(mn)$.

String shuffling

Let x, y, and z be strings. We want to know if z can be obtained only from x and y by interleaving the characters from x and y such that the characters in x appear in order and the characters in y appear in order. For example, if $x =$ **efficient** and $y =$ **ALGORITHM**, then it is true for $z =$ **effALGiORciIenTHMt**, but false for $z =$ **efficientALGORITHMextraCHARS** (miscellaneous characters), $z =$ **effALGORITHMicien** (missing the final t), and $z =$ **randomString** (obviously wrong). How can we answer this query efficiently? Your answer much be able to efficiently deal with strings such as $x =$ **aaaaaaaaaab** and $y =$ **aaaaaaaac**.