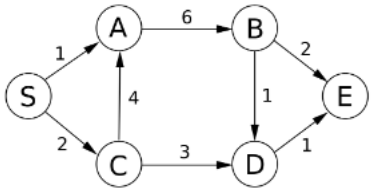
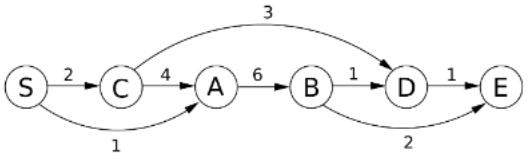
CS 170 DIS 06

Released on 2018-10-08

1 Shortest Paths with Dynamic Programming!





In this problem we will design a dynamic programming algorithm for finding the shortest $S - E$ path in a DAG like the one above.

(a) What are the subproblems you need to solve?

(b) Can you define the optimal solution to any subproblem as a function of the solutions to other subproblems?

(c) The dependency graph of a DP is a directed graph in which each subproblem becomes a vertex, and edge (u, v) denotes that subproblem u requires the solution to subproblem v in order to be solved. What does the dependency graph look like for this problem? What property of the dependency graph allows us to solve the problem using DP?

(d) If we proceed iteratively, what would be the best order to solve these subproblems?

(e) How many subproblems are there, and how long does it take to solve each? What is the overall running time of this algorithm?

(f) Run your algorithm on the example graph. What is the shortest path?

2 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 $z =$ **efficientALGORITHMS** (extra characters), $z =$ **effALGORITHMicien** (missing the final t), and $z =$ **effOALGRicieITHMnt** (out of order). How can we answer this query efficiently? Your answer must be able to efficiently deal with strings with lots of overlap, such as $x =$ **aaaaaaaaaab** and $y =$ **aaaaaaaac**.

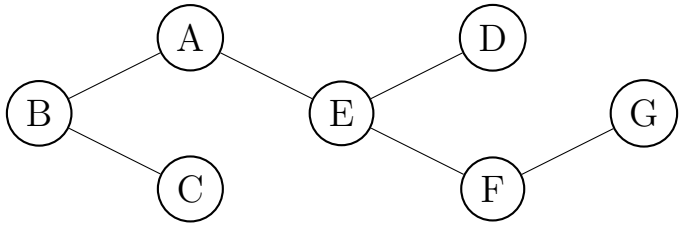
1. Design an efficient algorithm to solve the above problem and state its runtime.

2. Consider an iterative implementation of our DP algorithm in part (a). Naively if we want to keep track of every solved sub-problem, this requires $O(|x||y|)$ space (double check to see if you understand why this is the case). How can we reduce the amount of space our algorithm uses?

3 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 (covers) every edge $e \in E$. Give a dynamic programming algorithm which finds a vertex cover of a **tree** 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: $\{B, E, G\}$.



(Bonus: can you find a greedy algorithm for this problem?)