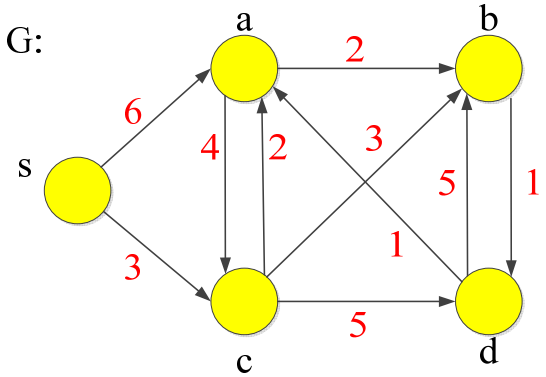
1. A dynamic programming algorithm solves an optimization problem by dividing the problem into subproblems. It then solves all subproblems that might be potentially needed. To avoid solving the same subproblems over and over, once the solution to a subproblem is computed, it is stored in a table (array). When a solution to a subproblem is needed, instead of recomputing the solution, the algorithm obtains it from the table. A dynamic programming algorithm computes the simplest subproblem first and then works its way up to the original problem. It usually relies on the concept of a recursive formula. Design a dynamic programming algorithm to find the longest common subsequence of two given sequences X and Y. Note that a subsequence of a sequence X is obtained by deleting 0 or more (not necessarily consecutive) symbols from X. For example, <A, A, B, C, D, F> and <F, A, B> are subsequences of X=<F, A, E, A, B, C, A, D, F>. (15%)

   Afterwards, based on the algorithm just designed, write an algorithm to find the longest monotonically increasing subsequence of a given sequence X. Note that <A, A, B, C, D, F> is a monotonically increasing subsequence of X=<F, A, E, A, B, C, A, D, F>, but <F, A, B> is not. In practice, <A, A, B, C, D, F> is the longest monotonically increasing subsequence of X. (10%)

2. As given below, G=(V, E) is a non-negative-weighted connected directed graph, where V is the set of nodes represented by circles and E is the set of edges represented by arcs. According to Dijkstra's shortest path algorithm, write an algorithm to generate G's shortest path tree T rooted at the specific node s, such that we can find a shortest path (i.e., the path with the minimum weight) from s to every other node along the tree. You should specify the parent node relationship between nodes of T in the algorithm. (15%)

   You should also demonstrate the detailed execution process of each iteration of your algorithm to generate the shortest path tree by putting within the circle the current minimum accumulated weight from the source to the associated node and by putting bold arcs from the parent node to the child node. Note that the accumulated weight is 0 for node s and infinite for every other node initially. (10%)



3. A sequence $Z = z_1, z_2, \ldots, z_k$ is a ***subsequence*** of $X = x_1, x_2, \ldots, x_n$ if there exists a strictly increasing sequence $\langle i_1, i_2, \ldots, i_k \rangle$ of indices of X such that for all $j = 1, 2, \ldots, k$, we have $x_{i_j} = z_j$, and if the index sequence is consecutive, i.e. $i_{j+1} = i_j + 1$ for $j = 1, 2, \ldots, k-1$, we call such a subsequence as ***substring***. Assume that the elements of the given sequence are integers. Consider the following two problems:
   [Problem A: Given a sequence X and an integer m, find a subsequence of X with the sum of its elements equal to m.]
   [Problem B: Given a sequence X and an integer m, find a substring of X with the sum of its elements equal to m.]
   Now determine which problem is NP-Complete and which is polynomial time solvable and give reasoning for your answer. (25%)
4. For the above two problems, Problem A and B, if the given integer m is small enough, then both problems can be solved efficiently. Design two such algorithms for the two problems. (25%)