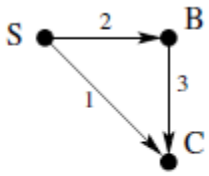
Quiz 2 – Practice Problems

1. Let G = (V, E) be a DAG, where each edge is annotated with some positive length. Let s be a source vertex in G. Suppose we run Dijkstra's algorithm to compute the distance from s to each vertex v ∈ V, and then order the vertices in increasing order of their distance from s. Are we guaranteed that this is a valid topological sort of G?

No .Justify your answer as follows: If you circled YES, then give one sentence that explains the main idea in a proof of this fact. If you circled NO, then give a small counterexample (a graph with at most 4 vertices) that disproves it.

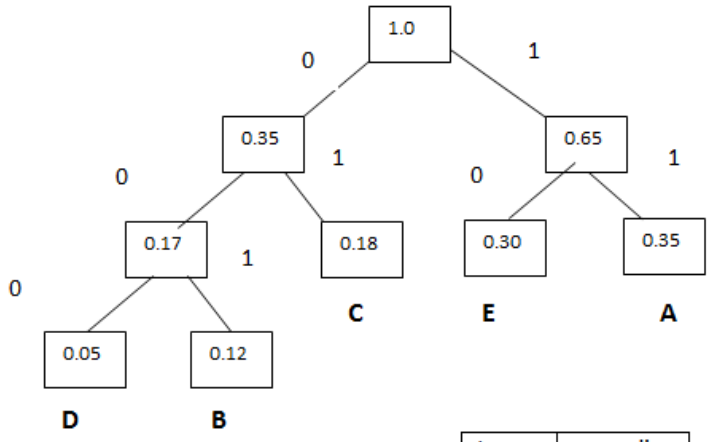
Answer:



Comment: Sorting by increasing distance gives S, C, B; but B must precede C in any valid topological sort.

2. Suppose we have an alphabet with only five letters A, B, C, D, E which occur with the following frequencies.

Letter	A	B	C	D	E
frequency	0.35	0.12	0.18	0.05	0.30



Letter	encoding
A	11
B	001
C	01
D	000
E	10

Quiz 2 – Practice Problems

3. Given a set $\{x_1 \le x_2 \le \ldots \le x_n\}$ of points on the real line, determine the smallest set of unit-length closed intervals (e.g. the interval [1.25,2.25] includes all x_i such that $\{1.25 \le x_i \le 2.25\}$) that contains all of the points. Give the most efficient algorithm you can to solve this problem, prove it is correct and analyze the time complexity.

The greedy algorithm we use is to place the first interval at $[x_1, x_1 + 1]$, remove all points in $[x_1, x_1 + 1]$ and then repeat this process on the remaining points.

Clearly the above is an $O(n)$ algorithm. We now prove it is correct.

Greedy Choice Property: Let S be an optimal solution. Suppose S places its leftmost interval at $[x, x + 1]$. By definition of our greedy choice $x \le x_1$ since it puts the first point as far right as possible while still covering x_1. Let S' be the scheduled obtained by starting with S and replacing $[x, x + 1]$ by $[x_1, x_1 + 1]$. We now argue that all points contained in $[x, x + 1]$ are covered by $[x_1, x_1 + 1]$. The region covered by $[x, x + 1]$ which is not covered by $[x_1, x_1 + 1]$ is $[x, x_1)$ which is the points from x up until x_1 (but not including x_1). However, since x_1 is the leftmost point there are no points in this region. (There could be additional points covered by $[x + 1, x_1 + 1]$ that are not covered in $[x, x + 1]$ but that does not affect the validity of S'). Hence S' is a valid solution with the same number of points as S and hence S' is an optimal solution.

Optimal Substructure Property: Let P be the original problem with an optimal solution S. After including the interval $[x_1, x_1 + 1]$, the subproblem P' is to find an solution for covering the points to the right of $x_1 + 1$. Let S' be an optimal solution to P'. Since, $\text{cost}(S) = \text{cost}(S') + 1$, clearly an optimal solution to P includes within it an optimal solution to P'.

4. Let T be a complete binary tree with n vertices. Finding a path from the root of T to a given vertex v in T using breadth-first search takes

a. O(n)
b. O(lgn)
c. O(nlgn)
d. O(logn)
e. $O(n^2)$

Quiz 2 – Practice Problems

5. A Hamiltonian path in a graph G=(V,E) is a simple that includes every vertex in V. Design an algorithm to determine if a directed acyclic graph (DAG) G has a Hamiltonian path. Your algorithm should run in O(V+E). Provide a written description of your algorithm including why it works, pseudocode and an explanation of the running time.

Solution:

Compute a topological sort and check if there is an edge between each consecutive pair of vertices in the topological order. If each consecutive pair of vertices are connected, then every vertex in the DAG is connected, which indicates a Hamiltonian path exists. Running time for the topological sort is O(V+E), running time for the next step is O(V) so total running time is O(V+E).

PSEUDOCODE:

HAS_HAM_PATH(G)

1. Call DFS(G) to computer finishing time v.f for each vertex v.
2. As each vertex is finished, insert it into the front of the list
3. Iterate each through the lists of vertices in the list
4. If any pairs of consecutive vertices are not connected RETURN FALSE
5. After all pairs of vertices are examined RETURN TRUE

6. Consider a connected weighted directed graph G = (V, E, w). Define the *fatness* of a path P to be the maximum weight of any edge in P. Give an efficient algorithm that, given such a graph and two vertices u,v ∈V, finds the minimum possible fatness of a path from u to v in G.

We can see that that fatness must be the weight of one of the edges, so we sort all edge weights and perform a binary search. To test whether or not a path with a fatness no more than x exists, we perform a breadth-first search that only walks edges with weight less than or equal to x. If we reach v, such a path exists.

If such a path exists, we recurse on the lower part of the range we are searching, to see if a tighter fatness bound also applies. If such a path does not exist, we recurse on the upper part of the range we are searching, to relax that bound. When we find two neighboring values, one of which works and one of which doesn't, we have our answer. This takes $O((V + E) \lg E)$ time.

Quiz 2 – Practice Problems

7. **Scheduling jobs intervals with penalties**:

For each $1 \le i \le n$ job j_i is given by two numbers d_i and, where d_i is the deadline and p_i is the penalty. The length of each job is equal to 1 minute. We want to schedule all jobs, but only one job can run at any given time. If job i does not complete on or before its deadline d, we should pay its penalty. Design a greedy algorithm to find a schedule which minimizes the sum of penalties.

Observation: We can assume that all jobs finish after n minutes.

Proof: Suppose not. So there is an empty minute starting at say $0 \le k \le n-1$ (where no job is scheduled) and there is a job j_i for some $1 \le i \le n$ which is scheduled some time after n minutes. If we schedule job j_i to start at minute k, then this can only be a better solution since everything remains same except j_i might now be able to meet its deadline.

We assign time intervals M_i for $1 \le i \le n$ where M_i starts at minute $i-1$ and ends at minute i. The greedy algorithm is as follows:

- **Arrange the jobs in decreasing order of the penalties $p1 \ge p2 \ge \cdots \ge pn$ and add them in this order**
- **To add job j_i ,**
  - **if any time interval M_l is available for $1 \le l \le d_i$, then schedule j_i in the last such available interval.**
  - **else schedule j_i in the first available interval starting backwards from Mn**

What is the running time of your algorithm? Explain.

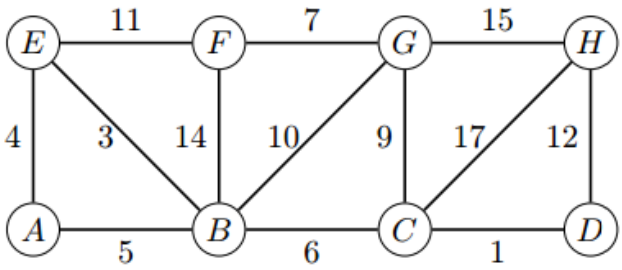
1. The sorting of by penalties takes Θ(nlgn)

2. To find the spot in the schedule can take in worst case 1+2+...+ n = O(n^2) depending on the data structure used this could be reduced so you can use Big O instead of theta.

Overall O(n^2) or Θ(n^2).

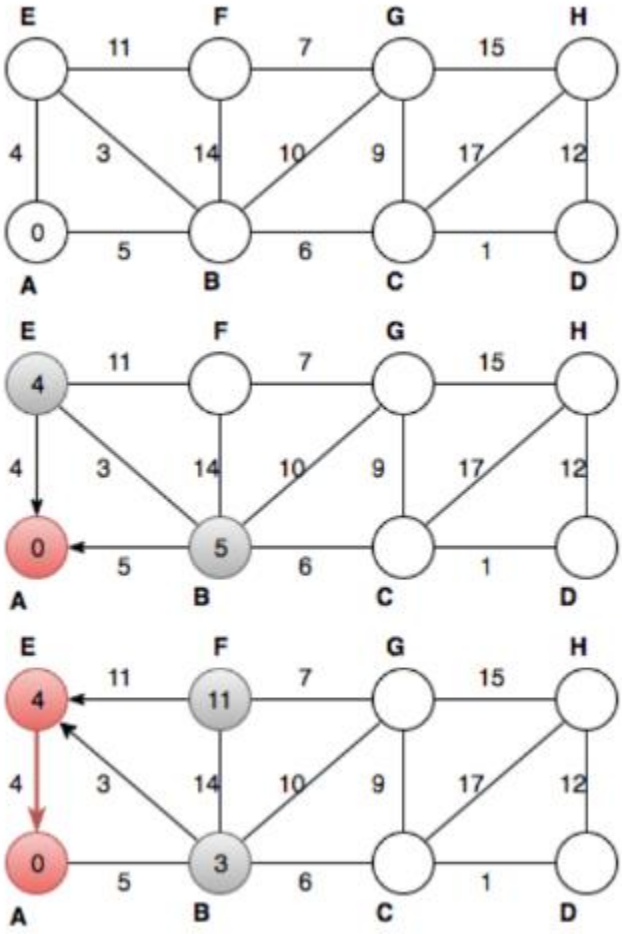
Quiz 2 – Practice Problems

8. Consider the weighted graph below:



(Demonstrate Prim's algorithm starting from vertex A. Write the edges in the order they were added to the minimum spanning tree.

Solution $(AE, BE, BC, CD, CG, FG, DH)$



Quiz 2 – Practice Problems