CS 260 – Homework #9

Chris Kasper

March 9, 2018

Problem 1 (6.2 from text)

Describe a mathematical model for the following scheduling problem. Given tasks $T1$, $T2$, . . . , Tn, which require times $t1$, $t2$, . . . , tn to complete, and a set of constraints, each of the form "Ti must be completed prior to the start of Tj," find the minimum time necessary to complete all tasks.

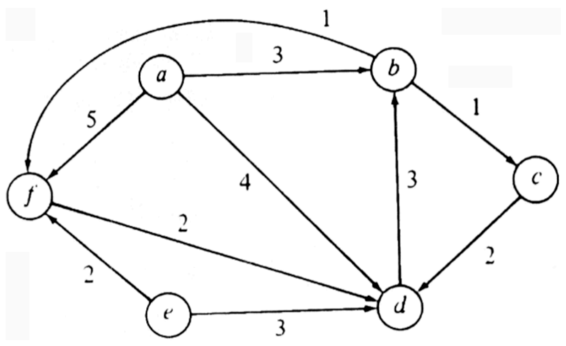
Solution

Given that we're are just trying to find the minimum amount of time needed to complete all the tasks, and that the tasks have constraints, we need to find the shortest path to the end that takes the longest amount of time.

Therefore, the minimum time necessary to complete all tasks is the summation of the max times of each "Ti that needs to completed before moving to Tj,".

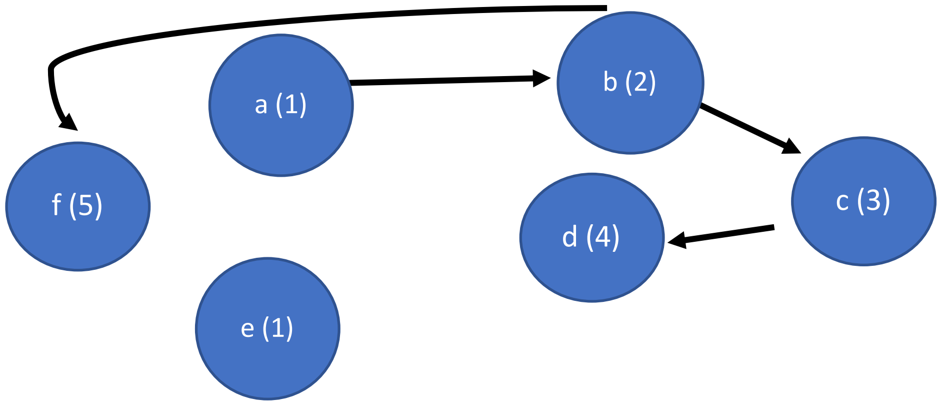
Problem 2 (6.8 from text)

Assuming the order of the vertices is a, b , . . . , f in Fig. 6.38, construct a depth-first spanning forest; indicate the tree, forward, back and cross arcs, and indicate the depth-first numbering of the vertices.



Solution

Spanning tree (2): a,b,c,d,f | e



Arcs:

1. Tree: ab, bc, cd, bf
2. Forward: ad, af,
3. Back: db
4. Cross: fd, ef, ed

Problem 3 (6.10 from text)

A *root* of a dag is a vertex r such that every vertex of the dag can be reached by a directed path from r. Write a program to determine whether a dag is rooted.

Solution

```
Function dag_root( G: graph ){
    For vertices in G:
        Reached = {}
        V = vertices
        While V not in Reached and not NULL:
            Insert(V, Reached)
            V = V.next()
        If size(Reached) == G.vertices:
            Return true
        Else:
            Return False
    Return False
}
```

Problem 4 (6.14 from text)

Write a program to find the longest path in a dag. What is the time complexity of your program?

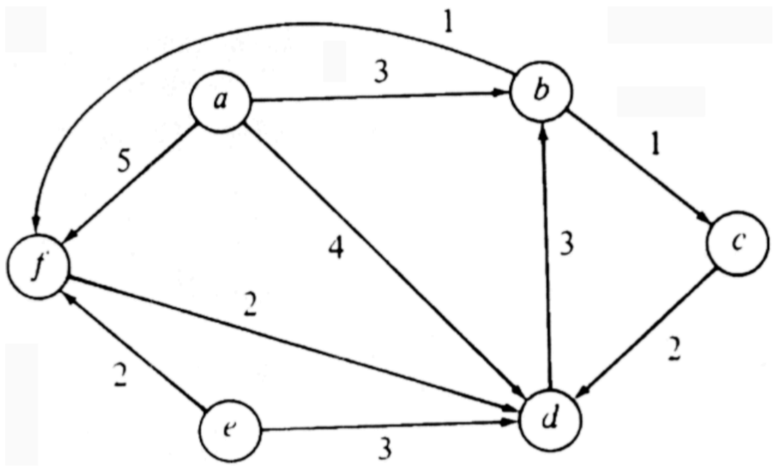
Solution

```
Function dag_longestpath ( G: graph ){
    FW(G) – $O(v^3)$
    # Look through distance matrix
    Longest_path = 0
    For v1 in G: - O(v)
        For v2 in G: - O(v)
            If d[v1,v2] > longest path:
                Longest_path = d[v1,v2]
    Return Longest_path
}
```

Time Complexity = $\max(v^3, v*v) \Rightarrow O(v^3)$

Problem 5 (6.15 from text)

Find the strong components of Fig. 6.38.



Solution

Strongly connected components:

1. a
2. e
3. b,c,d,f

Problem 6 (6.20 from text)

Write a program that takes as input a digraph and two of its vertices. The program is to print all simple paths from one vertex to the other. What is the time complexity of your program?

Solution

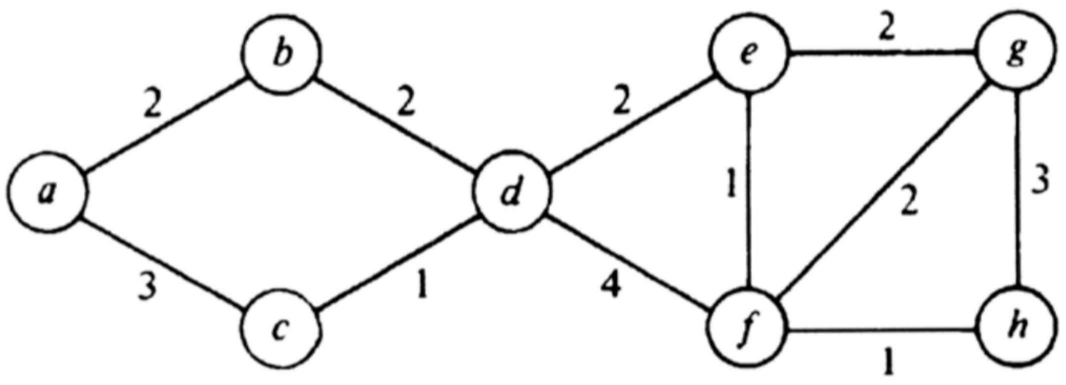
```
Function dag_simplepaths( G: graph, v1, v2: vertix ){
    Paths = {}
    v = v1
    #Not sure how to structure this loop below
    for loop to go through a vertex's children to find a new path: - (??)
        p = {}
        v = v1
        while v not NULL and not in p: - (??)
            if v == v2:
                Insert(v,p)
                Insert(p,Paths)
                Break
            Else:
                Insert(v,p)
            v.next()

    For p in Paths: - O(n)
        Print p
}
```

Time Complexity = $O(V+E)$?

Problem 7 (7.3c from text)

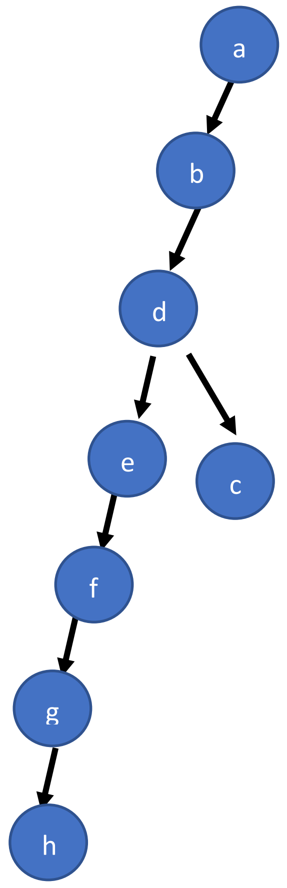
Consider the graph of Fig. 7.20.

c. Find a depth-first spanning tree starting at a and at d.



Solution

Starting at a:



Starting at d: