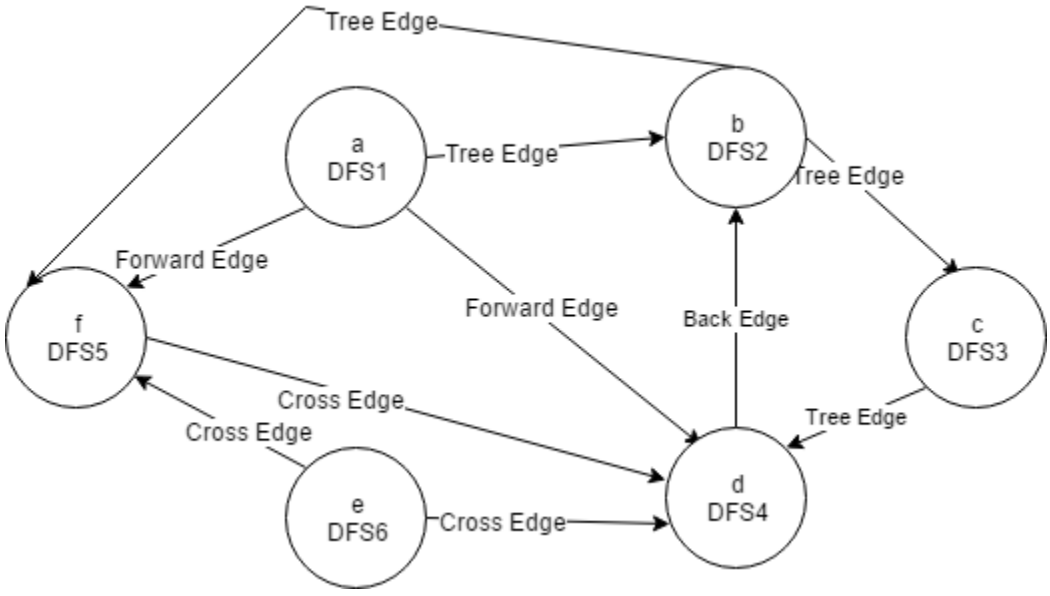
Steven Kravitsky

CS 260

Homework 9

1) You can create a directed graph with Tn verticies with tn weights from the starting vertex. This means that the first task will be adjacent to the starting vertex. You can calculate the minimum time by using an algorithm such as Floyd's to find the shortest path.



2)

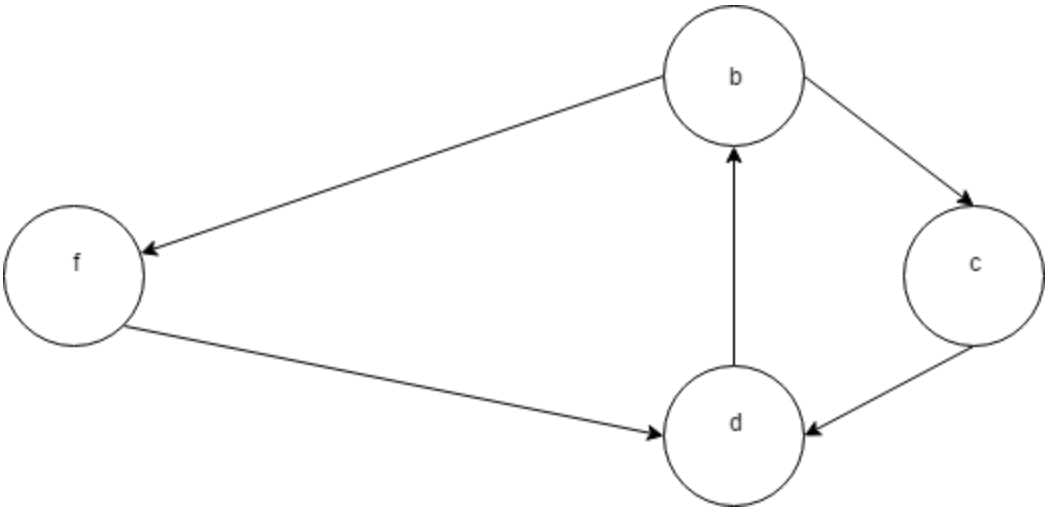
3)

```
Def DFS(G, V, M):
        M[v] = visited
        For w in adj[v]:
                If w is not in M
                        DFS(G, w, M)
        Return M
```

```
Def isRooted(G):
        For v in G:
                M = DFS(G, v)
                Rooted = True
                For x in G
                        If x is not in M
                                Rooted = False
                        If rooted is True
                                Return True
        Return False
```

4) Since this is a DAG we can negate edge weights and use Floyd's algorithm to find the shorted path. The algorithm is O(n^3) since that is the time complexity of Floyd's I,j,k loop.

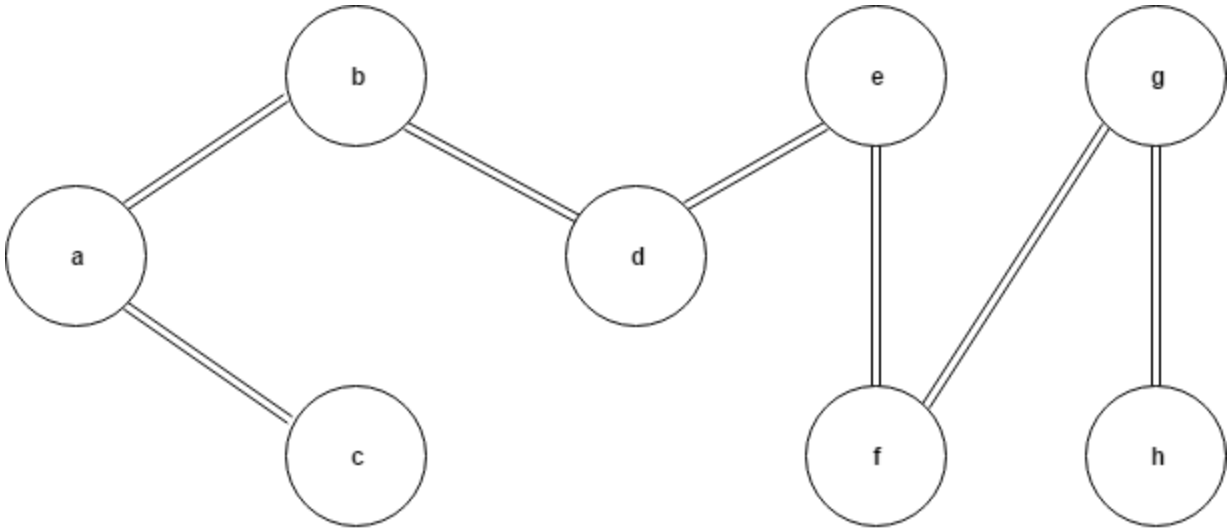
```
For I in dist       O(n^2)
        For j in dist
                dist[i][j] = -dist[i][j]

for k in G          O(n^3)
        for I in G
                for j in G
                        if dist[i][j] > dist[i][k] + dist[k][j]
                                dist[i][j] = dist[i][k] + dist[k][j]
                                parent[i][j] = k

max = dist[0][0]
I, j = 0
For x in dist       O(n^2)
        For y in dist[x]
                If max < dist[x][y]
                        Max = dist[x][y]
                        I = x
                        J = y
Path[]
While j is not None       O(n)
        P = parent[i][j]
        Path.ins(p,0)
        J = p
Print path
```

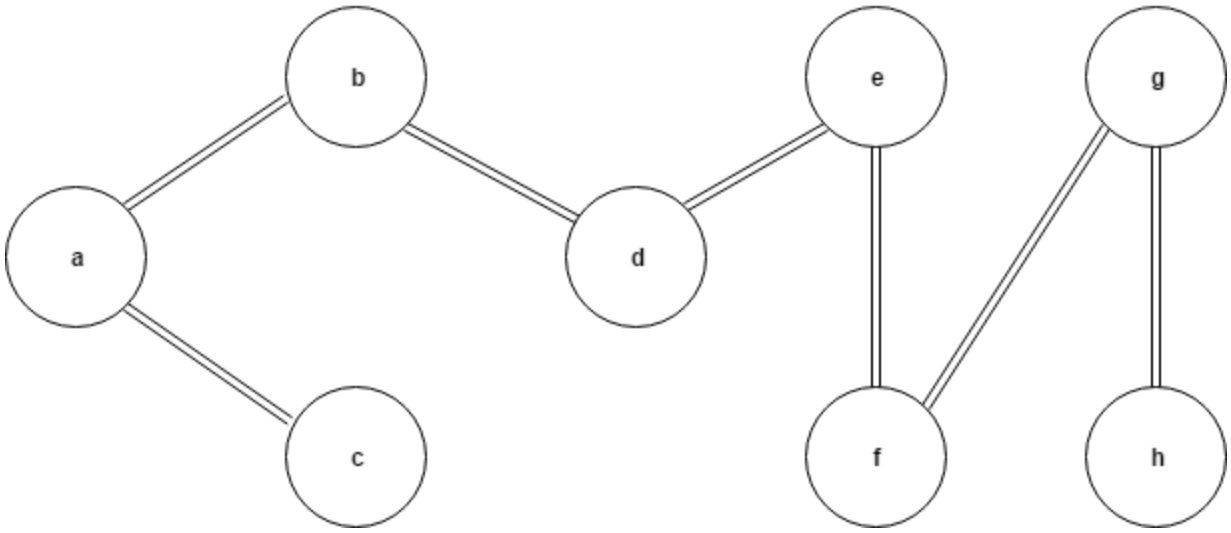


5)

6).

```
Def find_all(G, Start, End)
    Path = []
    Paths = []
    Queue = [(start, end, path)]
    While queue
            Start, end, path = Queue.pop()
            Print “PATH: ”, path
            Path = path + [start]
            If start == end
                    Paths.append(path)
            For x in set(graph[start]).difference(path)
                    Queue.append((node,end,path))
    Return paths
```





7)