Sean Morrissey

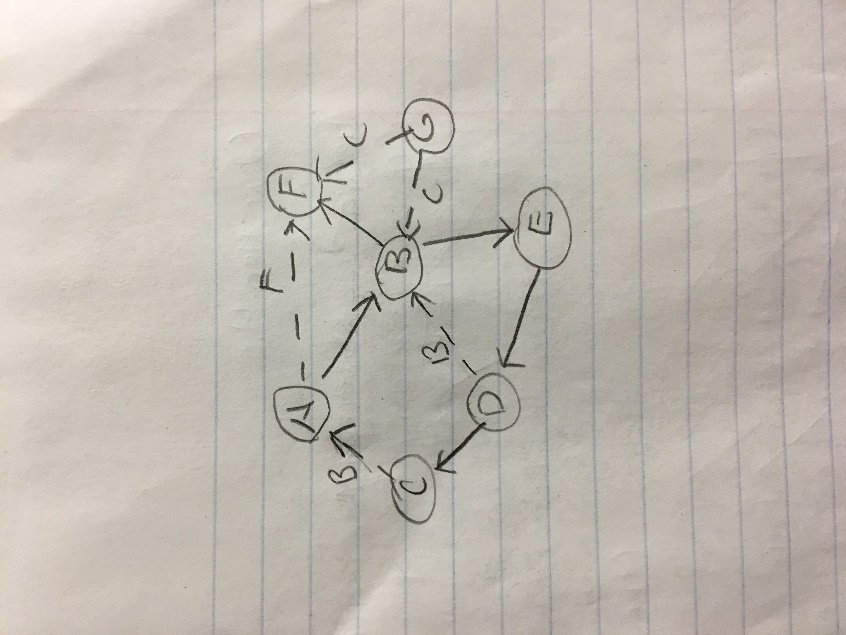
Professor Ramoza Ahsan

CS 2223 Algorithms

26 April 2019

Assignment 4

Question 1.



1. A (0,11, null)

B (1,10, A)

E (2, 7, B)

D (3, 6, E)

C (4, 5, D)

F (8, 9, B)

G (12, 12, null)

1. A (0, null)

B (1, A)

F (1, A)

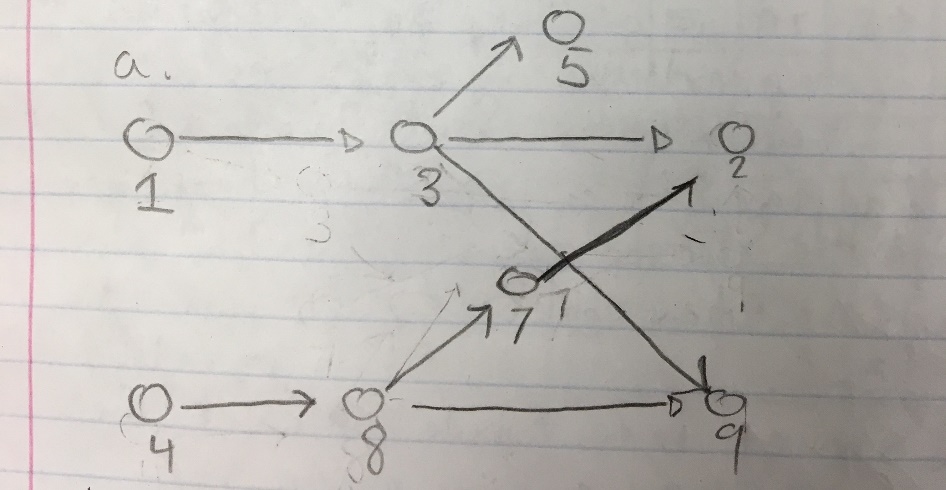
E (2, B)

D (3, E)

C (4, D)

G (5, null)

Question 2.



1. 1, 3, 5, 4, 8, 7, 2, 9

Question 3.

**PSEUDOCODE:**

reliablePath(u,v)

Given a directed graph G =(v,e)

w(u,v)

return (1/r(u, v))

Dijkstra (G, w, s):

Init (G, s)

S= null

Q = V[G]

While Q != null:

u = ExtractMin of Q

S = S union the set of u

For every vertex v that is an element of the adjacent graph of u (Adj[u]):

Relax (u, v, w)

**Analysis:**

For one to find the most reliable path between s and t, we us Dijkstra’s algorithm with edge weights w (u, v) equaling -, initializing the weights as such takes O(E) time. Because the most reliable path is the shortest path from s to t whose reliability is the product of the reliabilities of its edges, we can say that the runtime complexity is O (E+VlogV).

Question 4.

Create a pointer to a tree structure that points to null

Function findSum() takes a pointer to a tree node

If the node is null then

Return zero

Return the sum of the data at the node, findSum of the left side of the tree, and findSum

Of the right side of the tree

Function findMaxDiff() takes a pointer to a tree node

If the node is null then

Return zero

Vairable XL that is equal to findSum of the left sub tree of the node

Variable XR that is equal to findSum of the right sub tree of the node

If (XL – XR) is greater than a maxValue then

The maxValue is (XL-XR)

The maxNode is the current node

Call findMaxDiff on both left and right subtrees of the current node

Return the maxNode

**Analysis:**

findSum() calculates the sum of the nodes in the tree from the given node. This takes O(n) time where n is the # of nodes in the given tree. findMaxDiff() is O(n) time where n is the number of nodes in the tree. It is called for n nodes. Therefore, the overall complexity is O(n^2) time.

Question 5.

a.

Item: 1 2 3 4

Weight: 2 6 4 3

Value: 5 24 15 10

V/W: 2.5 4 3.75 3.33

The greedy knapsack algorithm would choose Item 2 and nothing else, having 1 kg left over and nothing to use it on

b.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I/W** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 5 | 5 | 5 | 5 | 5 | 5 |
| **2** | 0 | 0 | 5 | 5 | 5 | 5 | 24 | 24 |
| **3** | 0 | 0 | 5 | 5 | 15 | 15 | 24 | 24 |
| **4** | 0 | 0 | 5 | 10 | 15 | 15 | 24 | **25** |

The maximum value is $25, being a combination of item 4 and 3.

Question 6.

