Assignment 3

Wednesday, June 27, 2018 2:38 PM

COMP2611_COMP2000_S3_2017_2018_Assignment3[...

**COMP2611/COMP2000 – Data Structures**
**Semester 3, 2017-2018**
**Assignment 3**

**Graph** **Due: 11.55 p.m., July 7th, 2018**

Write a program to do the following:

(a) Read a description of a graph and create its adjacency list representation. The information at each node consists of a string no longer than 16 characters. Data for the program will be supplied as follows:

- The names of the nodes; each name is a one-word string consisting of letters and/or digits, for example, London. This portion of the data is terminated by the 'name' END. The number of nodes is unknown beforehand. The nodes are to be kept in a *binary search tree* (BST).
- The description of the edges of the graph. The edges emanating from a node are specified by the name of the node, followed by an integer, n, (indicating the number of edges leaving the node), followed by n pairs of values. Each pair consists of a node name (the child) and an positive integer weight (the cost of going from the node to the child). The number of edges is unknown beforehand. Edges from a node *must* be stored in *alphabetical* order by node name but they may be given in *arbitrary* order in the data. However, the *names* of the nodes are NOT to be stored in the adjacency list. Instead, the *BST location* of the node must be stored. Data is terminated by a node END.

Output the graph, listing the nodes in *alphabetical* order. Each node is followed by the name and weight of the edges leaving it (in alphabetical order).

(b) Read the names of two nodes and give the depth-first and breadth-first traversals of the graph from each of these nodes. Print only the names of the nodes.

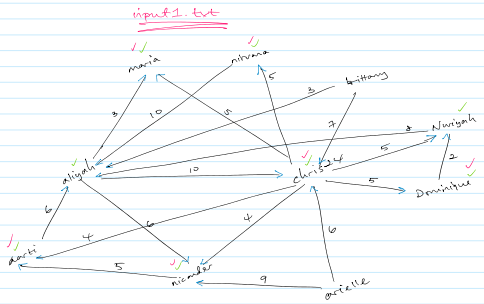
(c) Read the names of several nodes (one per line) and, for each node, find the minimal cost path to all the other nodes. Data is terminated by END. Use a *heap* structure to maintain the priority queue. Output the path and the minimal cost to each node. Use 9999 to represent an infinite cost.

**N.B.**

- Data are stored in the file **input.txt**.
- All output must be sent to the file **output.txt**.

**Submission Details:**

1. Submit one C or C++ file labeled with your ID number.
2. Include your full name, in comments, at the top of your program.
3. Email your submission to **sooklall7@gmail.com**.




Depth First Traversal.

Stack : christ

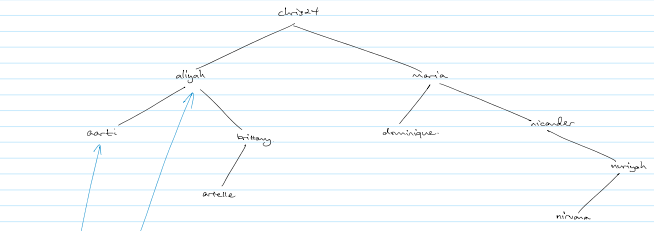
Result : christ aarti aligh maria nicander dominique nurayah artelle brittany

Can't read

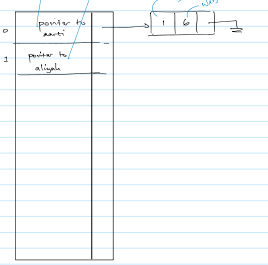
Breadth First Traversal.

Queue :

Result : christ aarti dominique maria nicander n




Could perform Binary Search on this list




Heap Design (Geeks for Geeks)

nirvana

nirvana nirvana nirvana

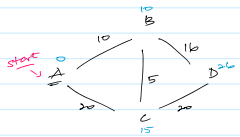
# Dijkstras

Saturday, June 30, 2018 1:59 PM

Youtube Geeks for Geeks.

Greedy Algorithm.

- Finds all possible routes & Find the Minimum Weight
- 2 Sets — ① A set of vertices where all paths are finalised
  - ② " " " " " Not finalised




Holds the Nodes that we still have to find shortest path for.

Put In Priority Queue (Min Heap)

START At 'A'

| END VERTEX | DISTANCE | Π Parent of End Vertex. |
|---|---|---|
| A ✓ | 0 | – |
| B ✓ | 10 | A |
| C ✓ | 15 | B |
| D ✓ | 26 | B |

'✓' = Removed from Priority Queue

THE ALGORITHM

(Array.

S = ∅ — A set of finalized vertices

priority Queue Q = V[i] — set of vertices where path is still to be found.

while Q != ∅

u = extractMin(Q) — Extract the vertex with shortest path from Q. (Always Extracts SRC first)

S = S ∪ {u} — Put u in set where the shortest path has been found.

for each v ∈ Adj[u] AND v ∈ Q. — loop stops when all edges from u are checked...

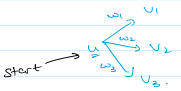
list of Edges from u

if d[v] > d[u] + w — distance from u→v. — distance from start to u.

currently known distance to v

d[v] = d[u] + w — New Shortest distance found.

π[v] = u — update parent.




TO FIND PARENT:

Perform Recursive Print with Delayed Output.

| pos | | | |
|---|---|---|---|
| 0 | A | 0 | – |
| 1 | B | ∞ | – |
| 2 | C | ∞ | – |
| 3 | D | ∞ | – |

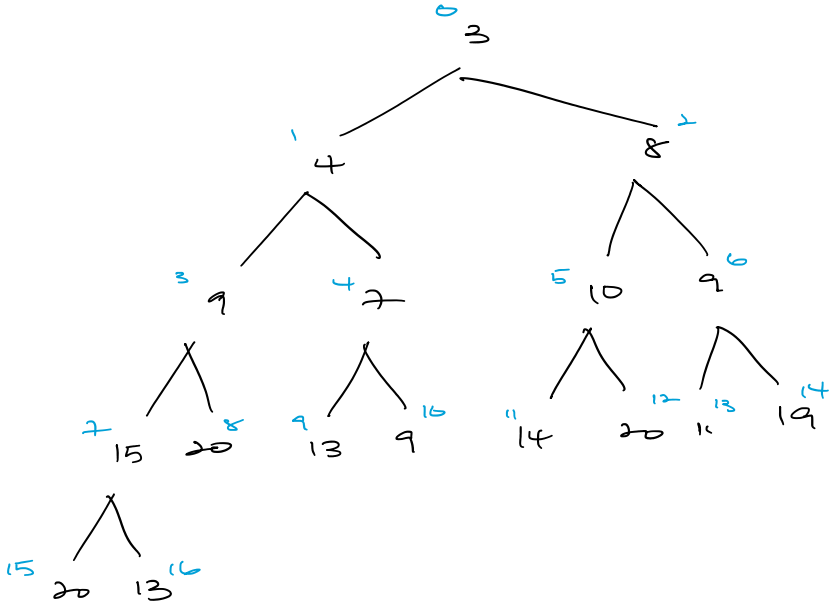
# Heaps

Saturday, June 30, 2018 12:39 PM

MIN HEAPS ≡ Priority Queues

Implemented using an ARRAY. A[V]

Array starts @ 0




PARENT (i - 1) / 2

INDEX

i

LEFT

i * 2 + 1

RIGHT

i * 2 + 2