CS566 Homework-4

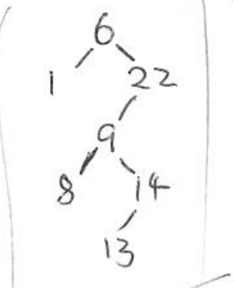
Due – Nov 15, 6:00 PM, at the class beginning

Student's Name: Jiankun Dong.

All work must be your own. Upload copy of your homework and any supporting materials (source code and test cases) online to Assignment4 in the CS566_A2 course site

No extensions or late submissions for anything other than major emergency. Show your work in detail.

Q1 [10 pts.]: Given the following key sequence (6,22,9,14,13,1,8) build the dynamic binary search tree WITHOUT BALANCING IT. How many probes (i.e., comparisons) does it take to determine that key 100 is not in the tree? (In this study case, the root is 6; the next element 22 > 6, so it goes to the right, etc).



① 6
② 22

$\boxed{\text{2 times}}$ first: 6<100 go right to 22
Second: 22< 100 go right to NIL (Null)

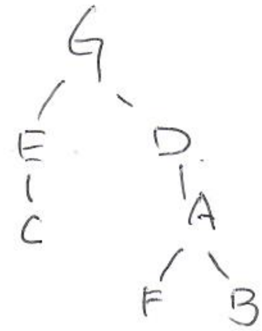
Q2 [15 pts.]: Find the depth-first search tree for the graph Fig.1 with G as the starting vertex. Assume that each adjacency list is in alphabetical order.

Q2:

Q3

(there's no space on next page...)

Reverse Alphabetic Order!



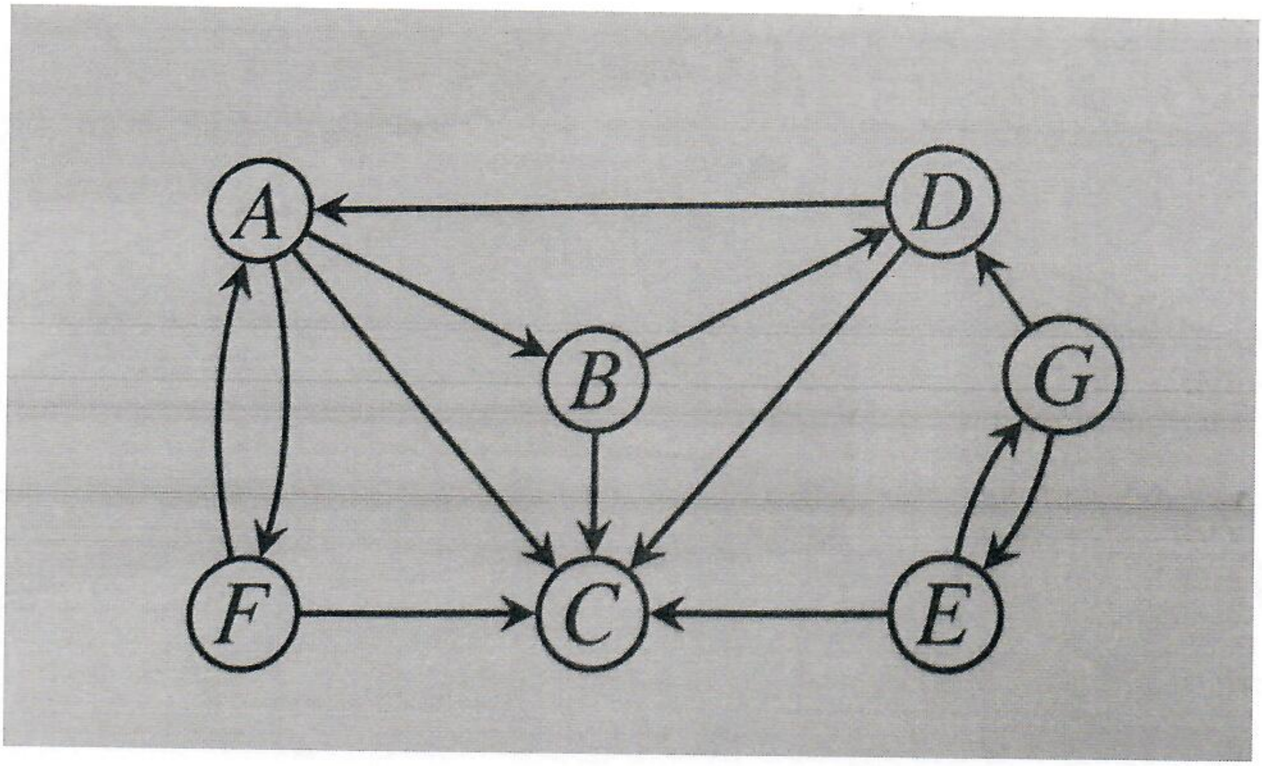
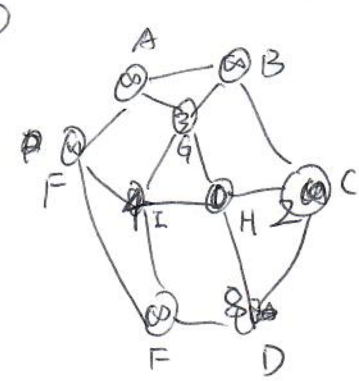


Fig.1

Q3 [15 pts.]: Find the breadth-first search tree for the graph Fig.1 with G as the starting vertex. Assume that each adjacency list is in reverse alphabetical order

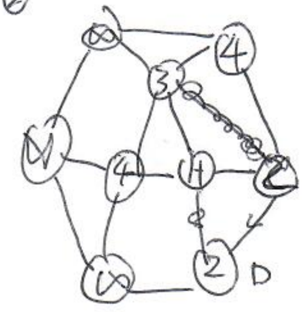
Q4 [20 pts.]: Execute Prim's minimum spanning tree algorithm by hand on the graph in Fig.2, showing how the data structures evolve. Clearly indicate which edge s become part of minimum spanning tree and in what order. Start at vertex H.

①



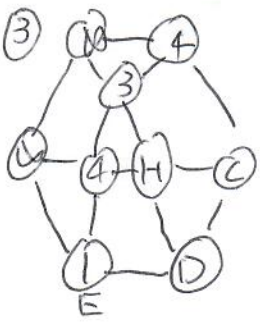
{H} + C

②



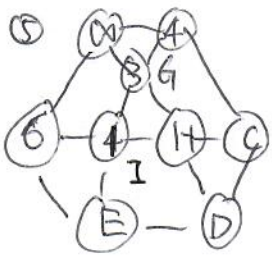
{H,C}+D

③



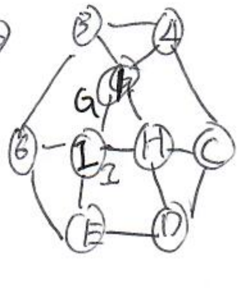
{H,C,D}+E

⑤



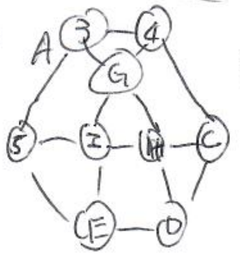
{H,C,D,E} + I

⑥



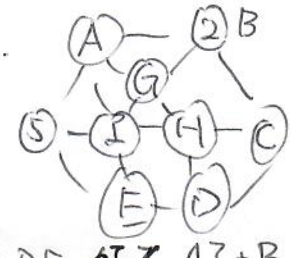
{H,C,D,E,I} + G

⑦



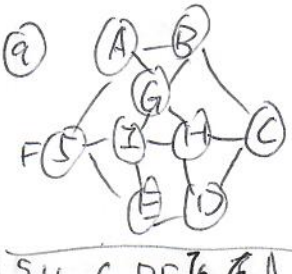
{H,C,D,E,I,G}+A

⑧



{H,C,D,E,I,G,A}+B

⑨



See next page about which edge gets added first

Final

{H, C,D,E,I,G,A,B,F}

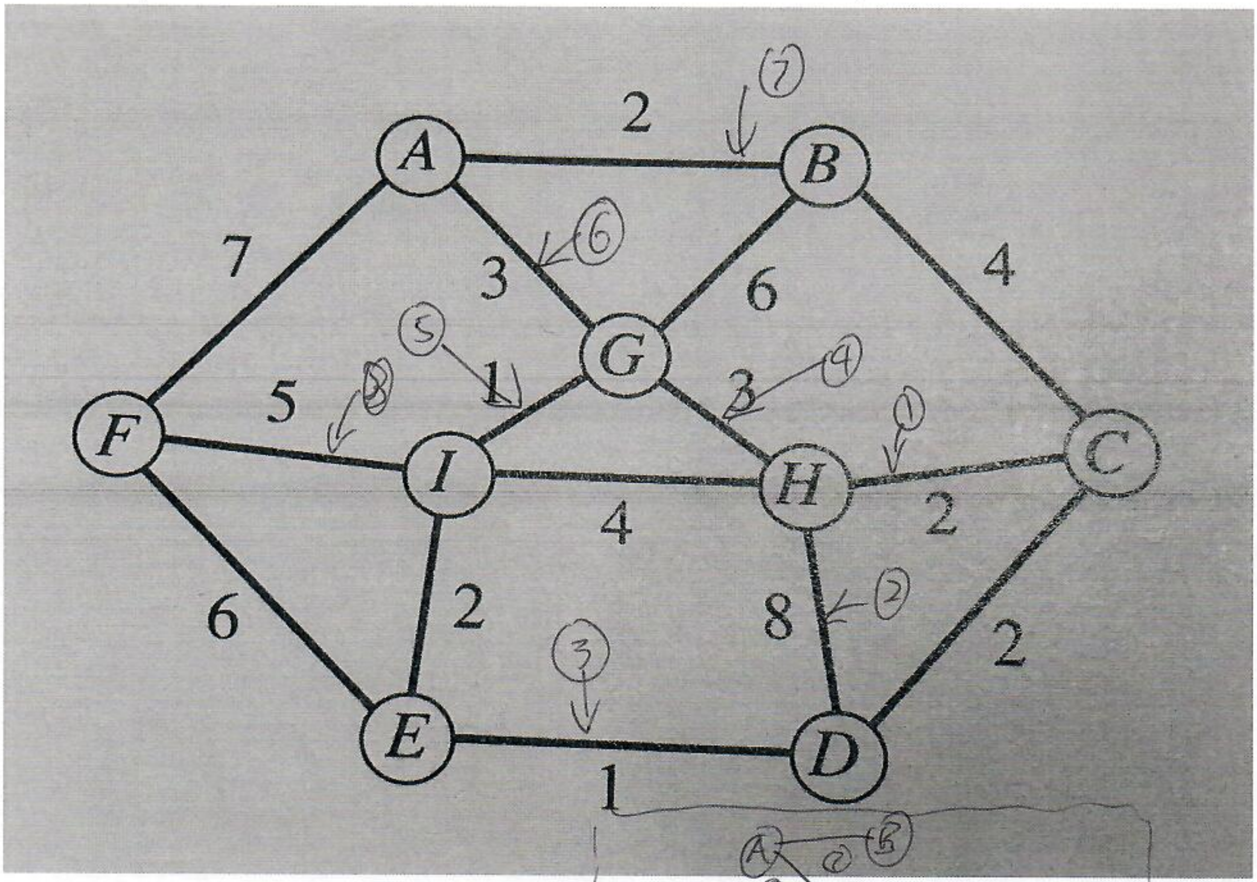
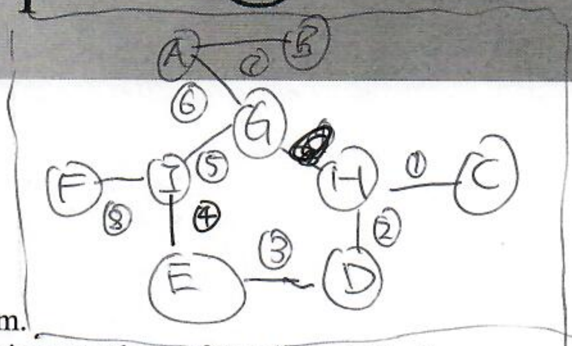


Fig.2

final & order:



Q5 [20 pts.]: Dijkstra's shortest path algorithm.
Here are the adjacency list with edge weights in parentheses for a digraph Fig.3

A: B (4.0), F (2.0)
B: A (1.0), C (3.0), D (4.0)
C: A (6.0), B(3.0), D(7.0)
D: A (6.0), E(2.0)
E: D (5.0)
F: D(2.0), E(3.0)

Execute Dijkstra's shortest-~~path-al~~gorithm by hand on this graph, showing how the data structures evolve, with s = A. Clearly indicate which edge become part of the shortest – path tree in what order.

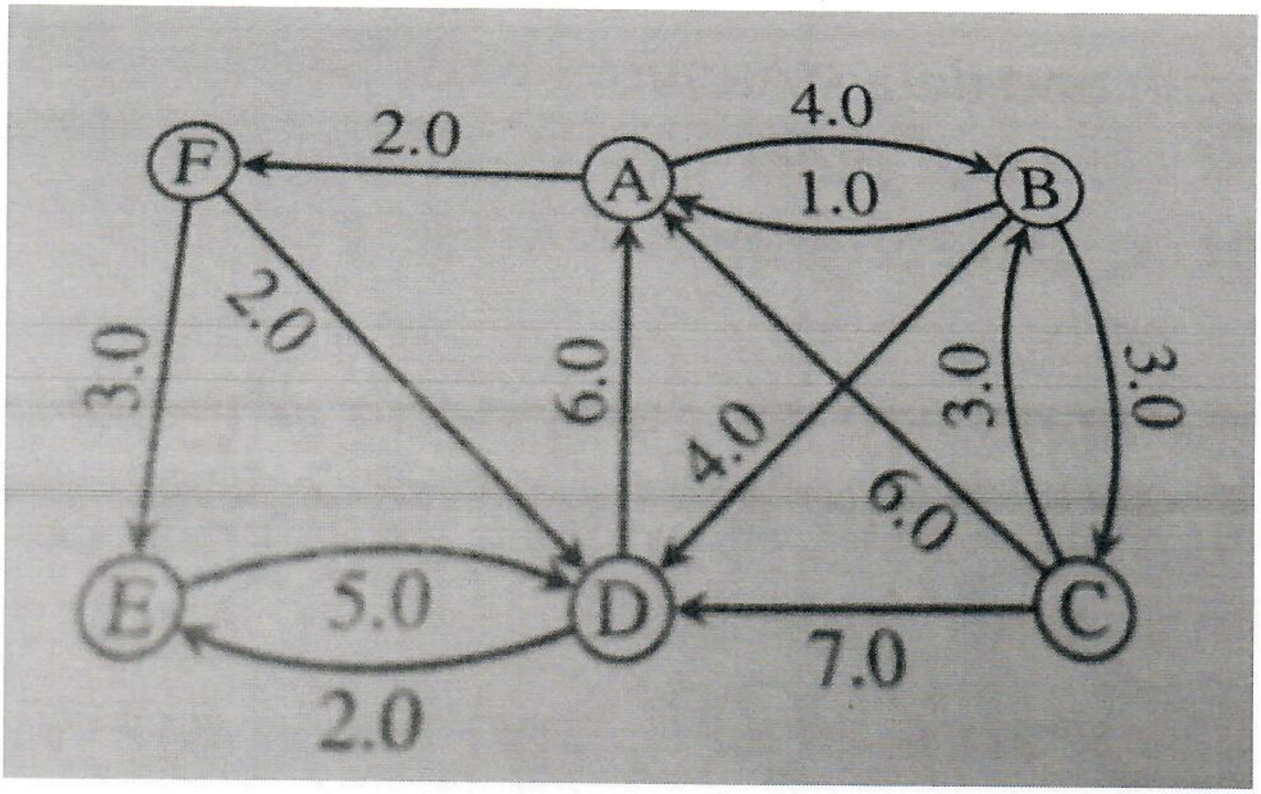


Fig.3

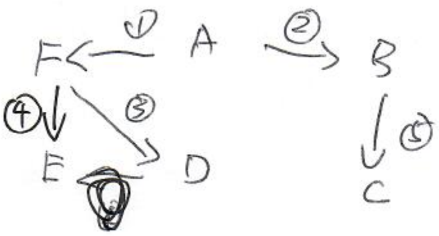
① min=2, A->F, {A,F}

② B=4 (AB) A→B, {A,F,B}
E = 5 (AFE)
D = 4 (AFD)

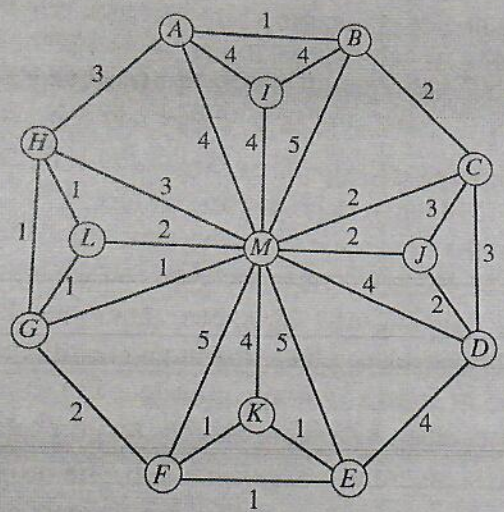
③ E=5 (AFE)
D=4 (AFD) or 8(ABD), F→D, {A,F,B,D}
C = 7 (ABC)

④. E = 5 (AFE) or 6(AFDE), F->E, {A,F,B,D,E}
C = 7 (ABC)

⑤ C = 7 (ABC), B->C, {A,F,B,D,E,C}



Q6 [20 pts.]: [Kruskal's minimum spanning tree algorithm]
Find the minimum spanning tree for the graph in Fig.4 that would be the output by Kruscal's algorithm, assuming that edges are sorted as shown on Fig.4



num = edge added w/order
X = Loop Detected.

1 2 3 4 5 6 7 8 9 10

Figure 8.14 Sorted edges: AB, EF, EK, FK, GH, GL, GM, HL, BC, CM, DJ, FG, JM, LM, AH, CD, CJ, HM, AI, AM, BI, DE, DM, IM, KM, BM, EM, FM.

11 12

Fig.4