Derrick DeBose

4.01)

1)

	A	B	C	D	E	F	G	H	I
fringe Wgt	0	3				7	2		
Parent	-1	A				A	A		
Status	(T)	F	U	U	U	F	F	U	U

	A	B	C	D	E	F	G	H	I
fringeWgt	0	3				7	2	8	1
Parent	-1	A				A	A	G	G
Status	T	F	U	U	U	F	(T)	F	F

	A	B	C	D	E	F	G	H	I
fringeWgt	0	3			5	5	2	5	1
Parent	-1	A			I	I	A	I	G
Status	T	F	U	U	F	F	T	F	(T)

	A	B	C	D	E	F	G	H	I
fringeWgt	0	3	4		5	5	2	5	1
Parent	-1	A	B		I	I	A	I	G
Status	T	(T)	F	U	F	F	T	F	T

	A	B	C	D	E	F	G	H	I
fringewgh	0	3	4	2	5	5	2	2	1
Parent	-1	A	B	C	I	I	A	C	G
Status	T	T	(T)	F	F	F	T	F	T

	A	B	C	D	E	F	G	H	I
FringeWgt	0	3	4	2	1	5	2	2	1
Parent	-1	A	B	C	D	I	A	C	G
Status	T	T	T	(T)	F	F	T	F	T

Derrick DeBose

4.01) 1 cont.)

	A	B	C	D	E	F	G	H	I
fringeWgt	0	3	4	2	1	5	2	2	1
Status	-1	A	B	C	D	I	A	C	G
Parent	T	T	T	T	T	F	T	F	T

	A	B	C	D	E	F	G	H	I
fringeWgt	0	3	4	2	1	5	2	2	1
Status	-1	A	B	C	D	I	A	C	G
Parent	T	T	T	T	T	F	T	T	T

	A	B	C	D	E	F	G	H	I
fringeWgt	0	3	4	2	1	5	2	2	1
Status	-1	A	B	C	D	I	A	C	G
Parent	T	T	T	T	T	T	T	T	T

2) The total weight of the MST is the summation of all the fringeWgt

$$\sum_{} \text{fringeWgt}(V) = 20$$

3) The lightest edge not in the MST is BG with the weight of 3. If this edge were added to the MST it would create a cycle AGB such that BG is the largest edge of the cycle.

The heaviest edge not in the MST is HD with the weight of 8. If this edge were added to the MST it would create a cycle CDH such that HD is the largest edge of the cycle.

>> Both cases break the MST Property
Proof by Contradiction

4.02) 1)

A	B	C	D	E	F
fringeWgt: d(A,A)=0 Parent: Status: (T)	d(A,B)=4 A F	U	U	U	d(A,F)=2 A F
fringeWgt: Parent: Status:	d(A,B)=4 A F	C U	D d(A,D)=4 F F	E d(A,E)=5 F F	F d(A,F)=2 A (T)
fringeWgt: Parent: Status:	B d(A,B)=4 A (T)	C d(A,C)=7 B F	D d(A,D)=4 F F	E d(A,E)=5 F F	F
fringeWgt: Parent: Status:	B	C d(A,C)=7 B F	D d(A,D)=4 F (T)	E d(A,E)=5 F F	F
fringeWgt: Parent: Status:	B	C d(A,C)=7 B F	D	E d(A,E)=5 F (T)	F
fringeWgt: Parent: Status:	B	C d(A,C)=7 B (T)	D	E	F

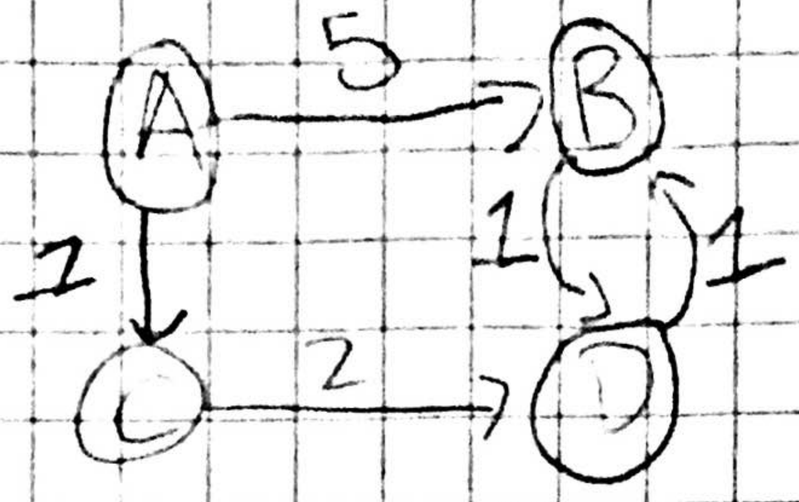
(T) - Added tree edge at that step

2) The shortest distance from (A,E) can be found in the fringeWgt of destination source E. d(A,E) = 5.

3) The shortest path from (A,D) can be found by starting at destination D in the priority queue and following the parent all the way up to the source vertex A. D's parent is F and F's parent is A so the path from (A,D) using the final priority queue is A → F → D.

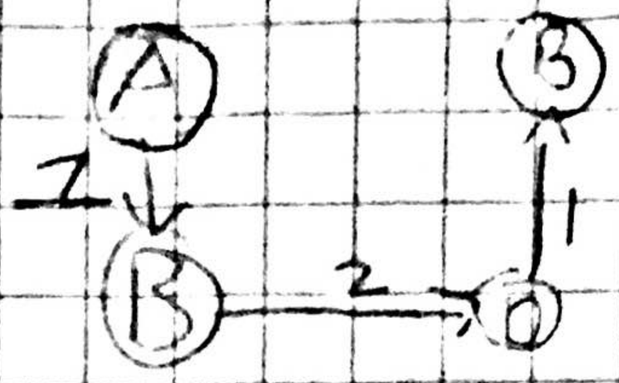
Derrick DeBord

41.03)



example graph

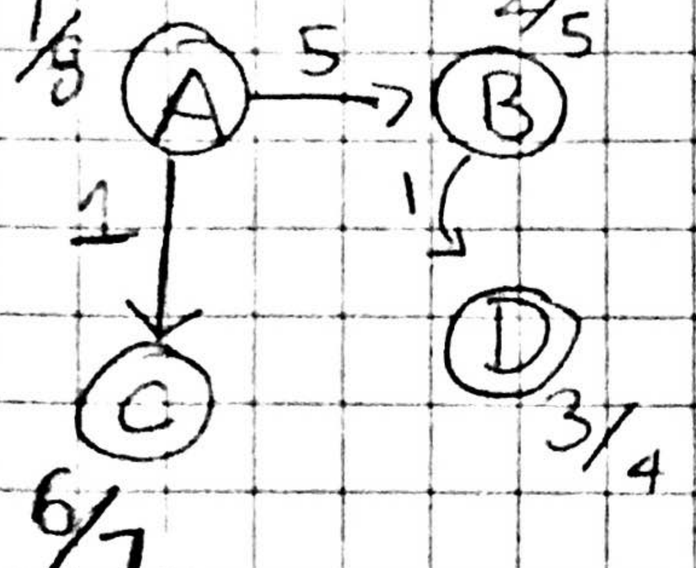
MST



optimal MST total: 4

Source vertex: A

DFS



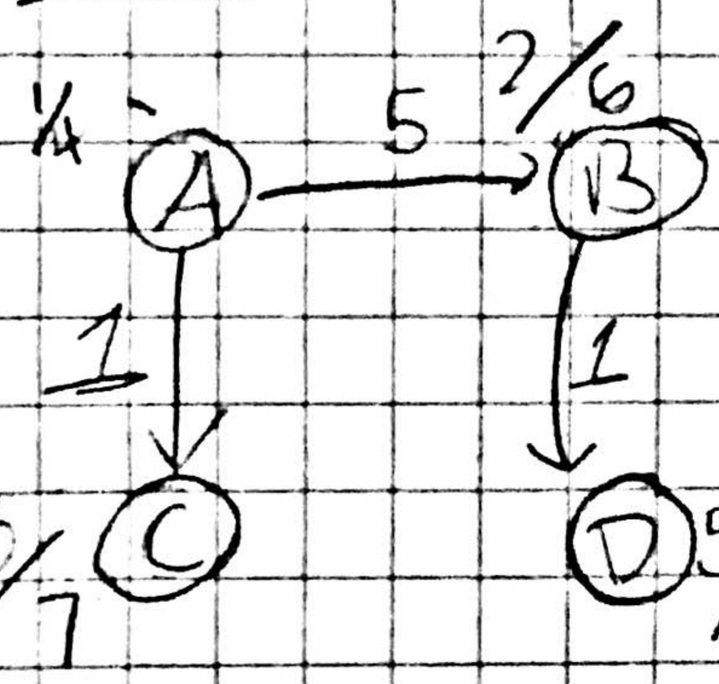
A
C
B
D

stack

top

DFS MST total: 7

BFS



Queue: A.
Queue: BC
Queue: CD
Queue: D
Queue: empty.

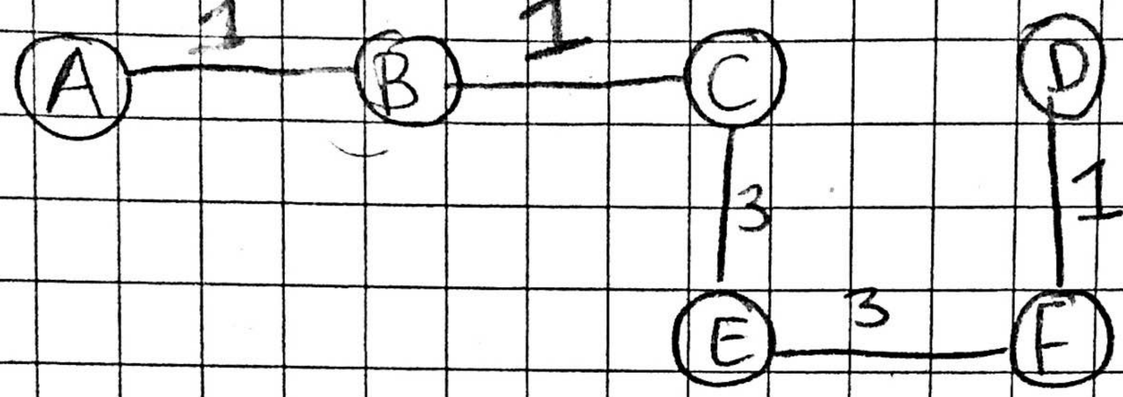
BFS MST total: 7

BFS & DFS doesn't find the shortest path tree because in a weighted graph, the algorithms are just finding some tree based on some pre-set ordering that give vertices priority over when making decisions.

BFS & DFS does not take into account the weights of the graph. The algorithms can find obtain an optimum solution based on the pre'set ordering scheme but doesn't guarantee a shortest path tree with weighted edges

Derrick DeBoe

4.04) Prim's MST

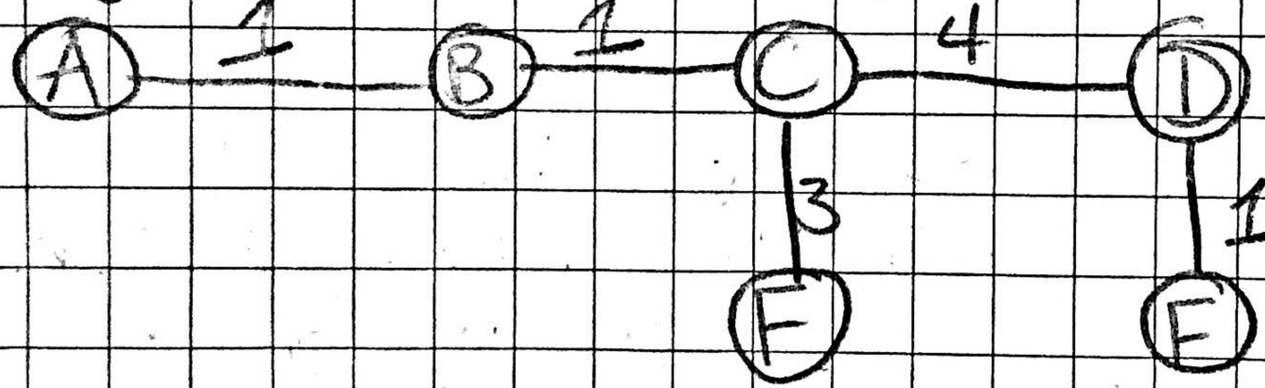


total weight = 9

d(A, F) = 8

d(A, D) = 9

Dijkstra's SSSP



total weight = 10

d(A, F) = 7

d(A, D) = 6

Derrick Dean

4.05) Implementation independent Dijkstra's will take time

$$T(n,m) = O(nT(getMin) + nT(deleteMin) + mT(decreaseKey))$$

we don't need to include insert because we can assume insert < deleteMin.

we know m can be $\geq n$ so we minimize decreaseKey runtime to $O(1)$ and increase the rest time of getMin and deleteMin to n

So the simplest implementation runs in

$$\Theta(n^2 + m) = \boxed{\Theta(n^2)}$$

Using a binary heap instead will reduce the running time because decreaseKey still takes $O(1)$ time but getMin and deleteMin will now only take $O(\log n)$ time.

So the binary heap implementation will take time

$$\Theta(m + n \log n)$$

The binary heap is the better implementation for planar graphs because m will not be asymtotricly larger than n in planar graphs. We can say n^2 is asymtotically larger than $n \log n$ so binary heap will be faster in most cases.

Derrick DeBose

4.06) Before Kruskal's algorithm starts choosing edges, it must first sort the edges in order from smallest to largest where the first edge added to the MST is the smallest edge in the MST.

The priliminary work takes O(log m) to build the min heap in which this operation can be called m times in the worst case.

The worst case for Kruskal's MST is $\Theta(m \log m)$. This case occurs when the largest edge in the graph is in the MST so log m will be called on all m edges.

The best case for Kruskal's MST is $\Theta(n \log m)$. This case occurs when all of the smallest edges are in the MST so log m will be called n-1 times because we know we have an MST when n vertices have n-1 edges.

Derrick DeBose

4.07) 1)

(A) (B) (C) (D) (E) (F) (G) (H) (I)

take DE

(A) (B) (C) (DE) (F) (G) (H) (I)

take GI

(A) (B) (C) (DE) (F) (GI) (H)

take GA

(GIA) (B) (C) (DE) (H) (F)

take CD

(GIA) (B) (DEC) (H) (F)

take CH

(GIA) (B) (DECH) (F)

take BA

(DECH) (GIAB) (F)

discard BG
take BC

(DECHGIAB) (F)

discard IE
take IF

(DECHGIABF)

Derrick DeBose

4.07) 2) The algorithm doesn't need to look at everyedge. It only has to find n-1 edges to consider in the best case because all MST have n-1 edges. It doesn't have to look past n-1 edges to be optimal because we know the remaining edges must be greater than or equal to the edges in the MST and by adding any of those edges we will be breaking the MST property of adding an edge creating a cycle in which the edge added is the largest edge of the cycle.

3) If BG edge weight of 3 is added to the MST then it creates a cycle (ABG) such that BG is the largest edge of the cycle.

If IH edge weight of 5 is added to the MST then it creates a cycle (IHCBAG) such that IH is the largest edge of the cycle.