Name: _____

3013 Algorithms Exam 2
Spring 2025

1. (2 points) A self-balancing binary tree is often defined such that the height difference between its left and right subtrees of *any* node is restricted. What is the usual strictness for a tree to be considered an AVL-balanced tree?

   A. No difference in height is permissible (must be exactly the same).

C

   B. The difference cannot exceed 2 levels.

   C. The difference cannot exceed 1 level.

   D. The difference cannot exceed the logarithm of the tree size.

2. (2 points) Unbalanced (simple) Binary Search Trees (BSTs) have which main advantage over complex self-balancing BSTs (like AVL or Red-Black Trees)?

   A. They always guarantee $O(\log n)$ operations without any rotations.

B

   B. They incur lower overhead in maintenance operations.

   C. They require fewer nodes to be fully balanced.

   D. They automatically handle concurrent insertions better.

3. (2 points) For a naive, unbalanced Binary Search Tree that does *not* apply any rotations to maintain balance, what is the worst-case time complexity for search, insert, or delete?

   A. $O(\log^2 n)$

B

   B. $O(n)$

   C. $O(1)$

   D. $O(n \log n)$

4. (2 points) Among self-balancing BSTs, the _____ tree often provides slightly faster insertion and deletion on average, whereas the _____ tree is more rigidly balanced, making it advantageous in read-heavy workloads. Select the correct pairing:

   A. (Splay Tree, Treap)

C

   B. (AVL Tree, Red-Black Tree)

   C. (Red-Black Tree, AVL Tree)

   D. (Segment Tree, Fenwick Tree)

5. (2 points) In rotations for balancing BSTs (like AVL or Red-Black), a single rotation costs $O(1)$. However, the total cost per insertion or deletion, considering possible traversals and rebalancing steps, can reach up to:

   A. $O(n)$

C

   B. $O(\sqrt{n})$

   C. $O(\log n)$

   D. $O(n \log n)$

6. (2 points) Which of the following statements about AVL Trees is true?

   A. Not every AVL tree is a BST, because rotations can violate the BST property.

B

   B. Every AVL tree is indeed a BST, just with stricter balance criteria.

   C. AVL trees cannot store duplicate values at any node.

   D. AVL trees are strictly less efficient than all Red-Black trees for searching.

7. (2 points) The maximum allowed balance factor (difference between left and right subtree heights) for any node in an AVL tree is:

   A. 2

B

   B. 1

   C. 0

   D. 3

8. (2 points) After an insertion in an AVL tree, a node ends up with a balance factor of -2. Which of the following best describes the required rebalancing rotation(s)?

A. Always a single left rotation on the root, regardless of subtree configuration.

D

B. No rotation needed; -2 is within acceptable limits.

C. A left rotation followed by a right rotation, in all cases.

D. It depends on whether the heavier subtree is left or right (LL or LR, RL or RR).

9. (2 points) Why might AVL trees often outperform Red-Black trees in workloads dominated by searches?

A. AVL trees can store more data in each node.

C

B. Red-Black trees don't support insertions in $O(\log n)$.

C. AVL trees maintain a stricter balance, leading to shallower average heights.

D. Red-Black trees require too much manual color-setting for queries.

10. (2 points) What is the worst-case height for an AVL tree storing n nodes?

A. $O(n)$

C

B. $O(\sqrt{n})$

C. $O(\log n)$

D. $O(\log \log n)$

11. (2 points) A graph representation that takes $O(V^2)$ space, even if the actual number of edges is very small, is most likely referring to:

A. Adjacency List

B

B. Adjacency Matrix

C. Edge List

D. Incidence Matrix

12. (2 points) Which of the following is a key benefit of using an adjacency matrix to represent a graph?

A. Rapid edge existence checks (constant-time lookup).

A

B. Minimizes memory usage for large but sparse graphs.

C. Ensures $O(1)$ BFS computations.

D. It can represent infinite graphs trivially.

13. (2 points) An adjacency matrix is particularly suitable for:

A. Dense graphs, where $E \approx V^2$.

A

B. Sparse graphs, where $E \ll V^2$.

C. Directed Acyclic Graphs exclusively.

D. Hypergraphs with infinite edges.

14. (2 points) Using an adjacency list can be much more space-efficient in graphs that are:

A. Extremely dense with nearly all possible edges.

B

B. Sparse, where E (number of edges) is much smaller than V^2.

C. Fully bipartite with no cycles.

D. Above 1 million nodes regardless of edges.

15. (2 points) In which scenario does using an adjacency list really shine for performance?

A. When you rarely need to explore neighbors.

B

B. When you frequently iterate over all neighbors of a given vertex.

C. When you need constant-time checks for whether any two vertices are connected.

D. When you need to represent a complete graph without wasted space.

16. (2 points) Which of the following is another strong reason to prefer an adjacency list?

A. When the graph is sparse and we want to avoid $O(V^2)$ storage.

B

B. We want to store infinite edges in constant space.

C. We must store the edges in a sorted array for quick merges.

D. We need only partial adjacency information.

17. (2 points) A graph that can be divided into two disjoint sets U and V such that no edge exists between vertices within the same set is called:

D

A. An Eulerian Graph
B. A Directed Acyclic Graph
C. A Complete Graph
D. A Bipartite Graph

18. (2 points) Which algorithmic approach allows us to detect if a graph is fully connected, contains cycles, or can be colored bipartitely (2 colors)?

A

A. Using Depth-First Search (DFS) or Breadth-First Search (BFS), incorporating a color or parent-tracking scheme.
B. Sorting the adjacency matrix by degrees.
C. Randomized primality tests on the graph's edges.
D. Union-Find without path compression.

19. (2 points) Which data structure is often discussed in algorithms (e.g., for improving Dijkstra's complexity) but rarely fully implemented in production due to its complexity and constant factors?

B

A. Binomial Heap
B. Fibonacci Heap
C. Pairing Heap
D. Skew Heap

20. (2 points) What is a primary limitation of Dijkstra's Algorithm in its standard form?

C

A. It cannot handle directed edges.
B. It fails to find the shortest path in undirected graphs.
C. It does not support negative edge weights.
D. It runs in exponential time regardless of the graph size.

21. (2 points) AVL Trees guarantee that:

A

A. Our searches will be in O(lg n)
B. Our searches will be in O(n)
C. No Tree guarantees anything
D. All search values will exist

22. (2 points) You can use an AVL tree to sort data if you insert all the values and then:

D

A. Don't use floating point data
B. Use a Postorder traversal to print the data in sorted order
C. Use a PreOrder traversal to print the data in sorted order
D. None of the Above

23. (2 points) What would the worst case running time of your AVL sort be?

C

A. $O(lg\ n)$
B. $O(n)$
C. $O(n\ lg\ n)$
D. $O(n^2)$
E. $O(n^2\ lg\ n)$

24. (2 points) Binary Trees and Binary Search Trees:

B

A. Are exactly the same thing
B. One implies a sorted order, the other does not
C. One has a maximum of two children, and a minimum of zero
D. One has no leaves
E. BT is actually a referenced subset of BST and implies a total ordering

25. (2 points) Depth First Search attempts to search _____ in the graph and uses a _____ to assist with this endeavor.

D

A. Locally , Stack
B. Deep , Queue
C. Locally , Queue

D. Deep , Stack

26. (2 points) The time complexity for performing a graph traversal is _____:

C

A. $O(V - E)$
B. $O(V\ lg\ E)$
C. $O(V + E)$
D. $O(V * E)$

27. (2 points) What are the maximum number of edges that an undirected graph can contain?

A

A. $\frac{n*(n-1)}{2}$
B. $n * (n - 1)$
C. n^2
D. $V * E$

28. (2 points) Dijkstra's algorithm finds the shortest path from a source "S" to _____?

B

A. The "destination" node
B. All other nodes in the Graph
C. All pairs
D. All of the Above
E. None of the Above

29. (2 points) If an algorithm makes choices based on no other information besides what is known currently, it is known as a _____ algorithm.

B

A. Recursive
B. Greedy
C. Backtracking
D. Brute Force
E. Divide and Conquer

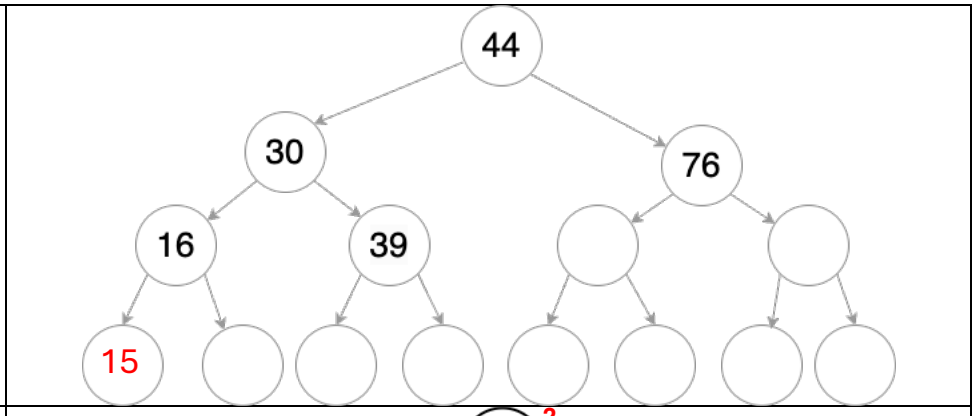
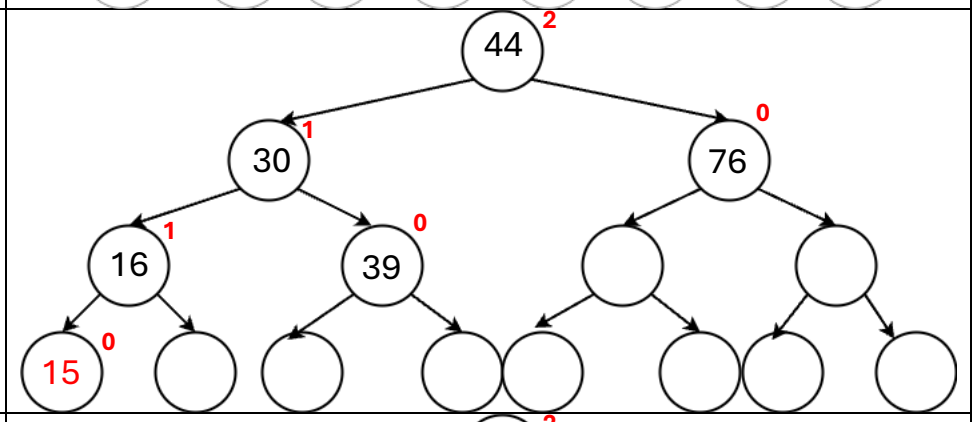
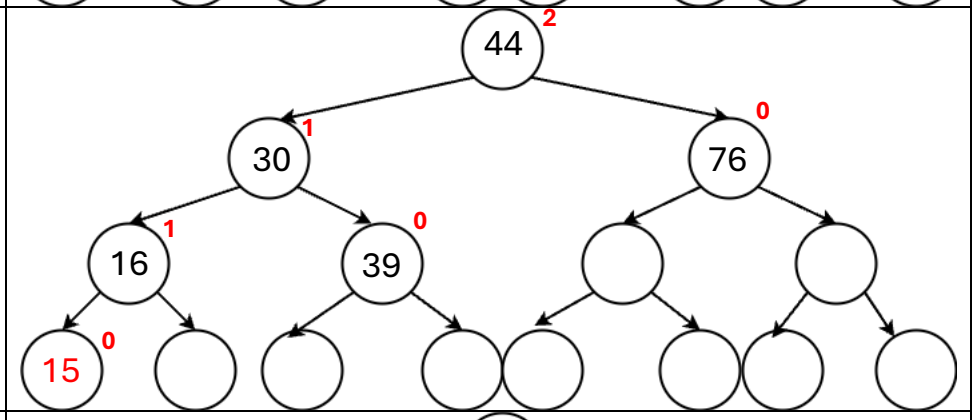
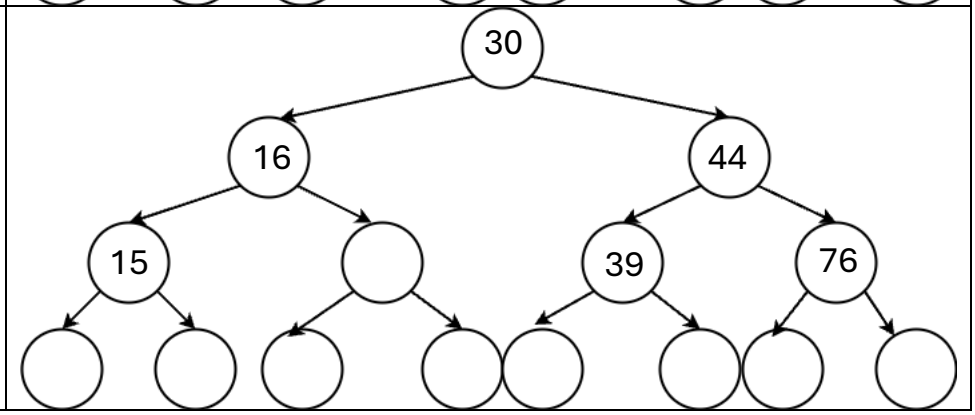
30. (2 points) Prims is good for _____ and Kruskels is good for _____ graphs.

A

A. Dense, Sparse
B. Dense, Dense
C. Sparse, Dense
D. Sparse, Sparse
E. None of the Above

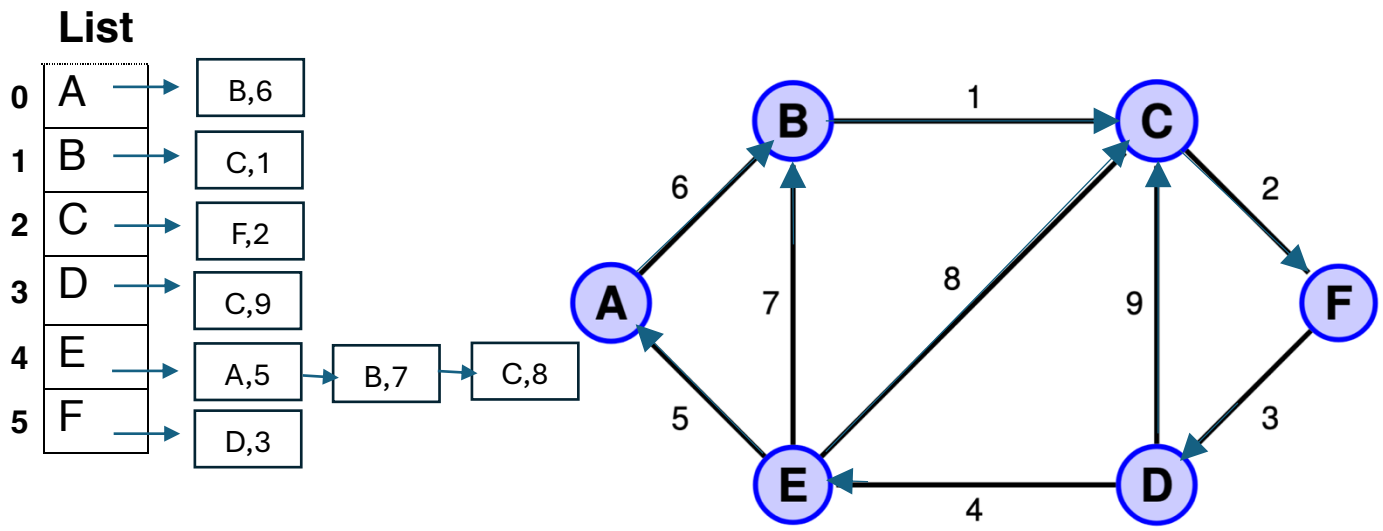
1) AVL Insert 15

Insert the value 15 into this AVL tree and apply any alterations necessary to maintain the rules of an AVL tree. Make sure you

Step 1: Insert 15 just like a regular BST.	
Step 2: Calculate AVL (rotation) values.	
Step 3: Right Rotation needed at tree root. - 30 becomes SubRoot - 44 left takes over 30's right child (39) - 30 Takes 44 as new right child - 30 Becomes new root	
Step 4: Done	

2) Graph Representations

Show a list based and matrix-based representation of the following graph.

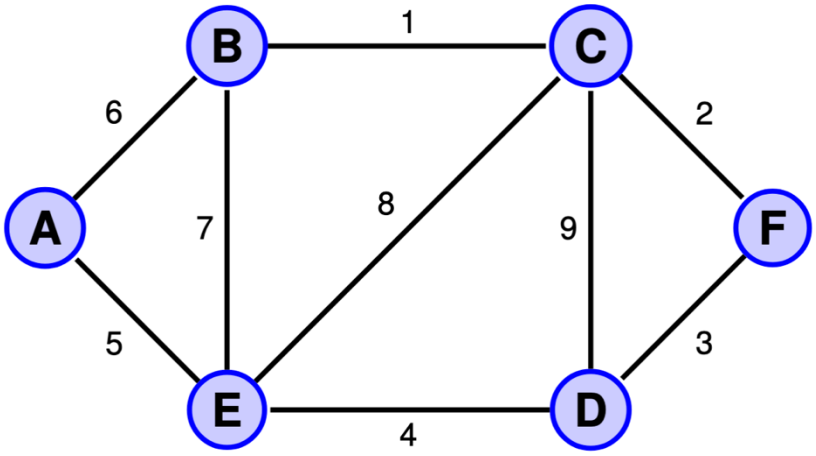


Matrix

	0 A	1 B	2 C	3 D	4 E	5 F
0 A		6				
1 B			1			
2 C						2
3 D			9		4	
4 E	5	7	8			
5 F				3		

4) DFS and BFS

Perform a depth first search and a breadth first search using the graph on this page and placing the vertex values (A-L) in the spaces provided below as they become visited.



DFS

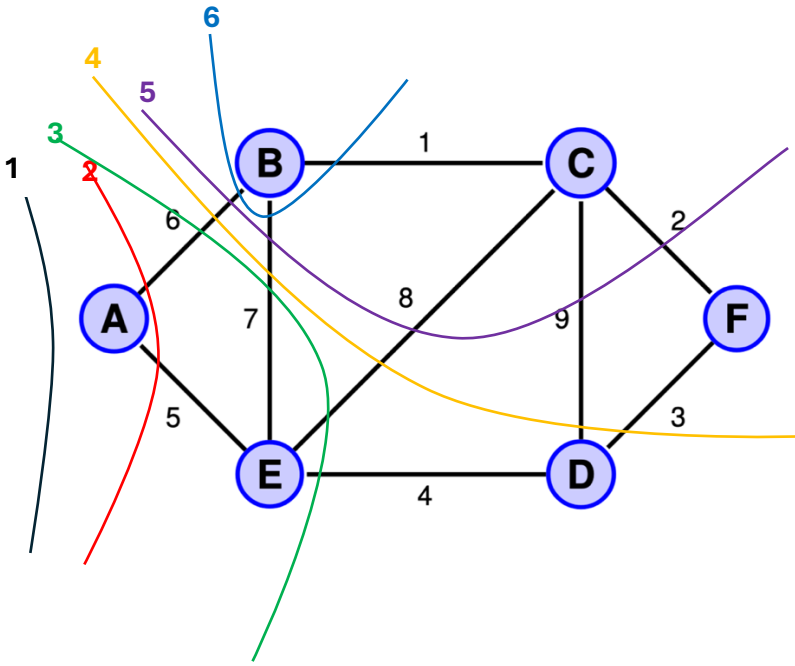
A	E	D	F	C	B	-	-	-	-	-	-
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**

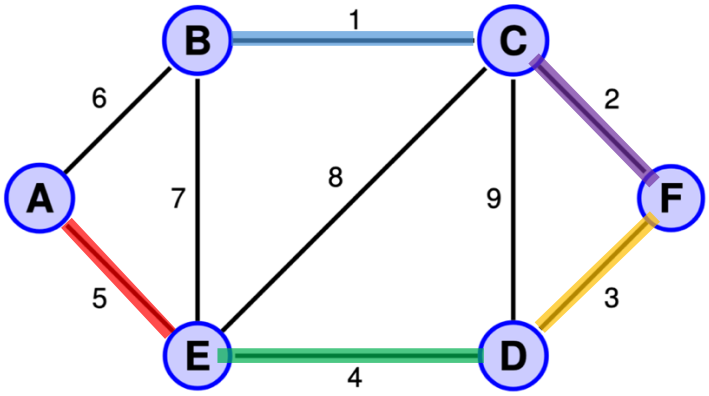
BFS

A	B	E	C	D	F	-	-	-	-	-	-
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**

5) MST: Prims

Add edges to the list on the right as they are added to the MST. Graph edges are typically written in a similar fashion to this: ***(Start Vertex, End Vertex, Edge Weight).*** Example: **D to F = (D, F,17).** Remember when making choices (if it applies here) do them alphabetically.



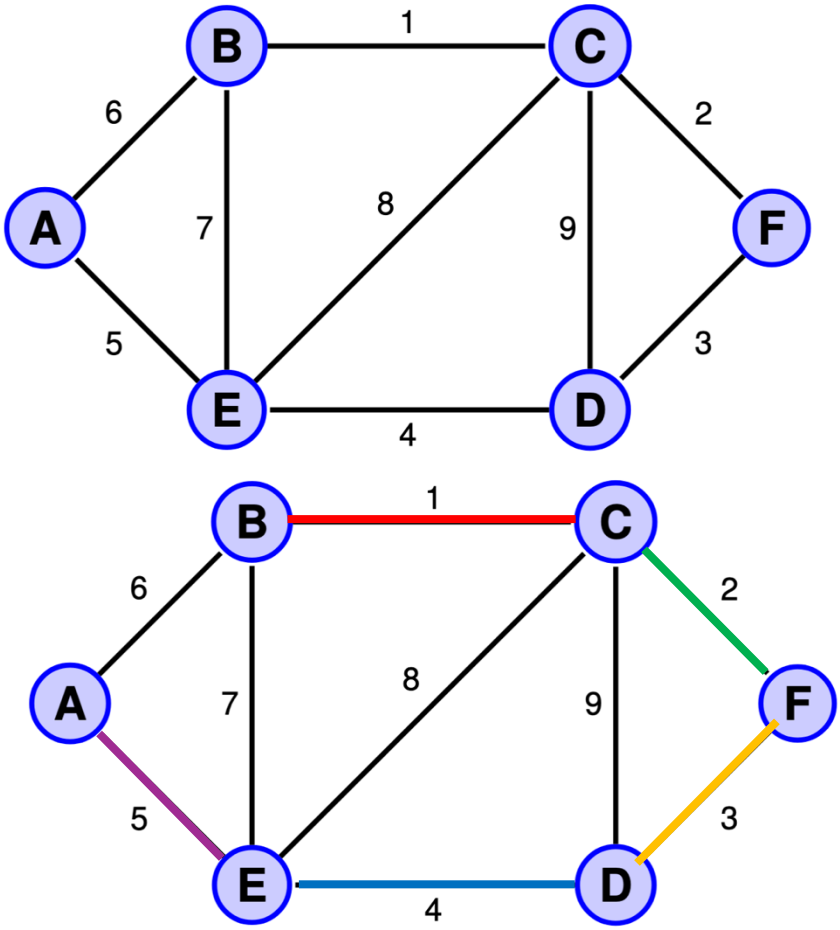


	CUT	Edges Crossed	MIN	MST
1	{ABCDEF}			A
2	{A} I {BCDEF}	(AB6) (AE5)	A,E,5	AE
3	{AE} I {BCDF}	(AB6) (BE7) (CE8) (DE4)	D,E,4	ADE
4	{ADE} I {BCF}	(AB6) (BE7) (CE8) (CD9) (DF3)	D,F,3	ADEF
5	{ADEF} I {BC}	(AB6) (BE7) (CE8) (CD9) (CF2)	C,F,2	ACDEF
6	{ACDEF} I {B}	(AB6) (BC1)	B,C,1	ABCDEF

6) MST: Kruskal's

Add edges to the list on the right as they are added to the MST. Refer to previous question for an example. Remember when making choices (if it applies here) do them alphabetically.

Priority Queue
~~(B,C,1)~~
~~(C,F,2)~~
~~(D,F,3)~~
~~(D,E,4)~~
(A,E,5)
(A,B,6)
(B,E,7)
(C,E,8)
(C,D,9)



Edges in MST
(B,C,1)
(C,F,2)
(D,F,3)
(D,E,4)
(A,E,5)
DONE

7) Shortest Path – Dijkstra's

Priority Queue
~~(AE – 5)~~
~~(AB – 6)~~
~~(BC-7)~~
~~(ED-9)~~
(EB-12)
(EC-13)

A	B	C	D	E	F
Visited Vertices					
A	E	B	C	D	

Previous	
A	
B	A
C	~~E~~ B
D	E
E	A
F	C

Distance Table					
A	B	C	D	E	F
∞	∞	∞	∞	∞	∞
0	6	13	9	5	9
		7			

1. Start with A
2. Add AB,AE to queue
3. Update distance to B & E
4. Choose shortest distance (E)
5. Update distances from E
6. …
7. …
8. …
9. …
10. …
11. …
12. …