CSC 017: Fundamentals of Computer Science III: Advanced Data Structures and Object-Oriented Programming

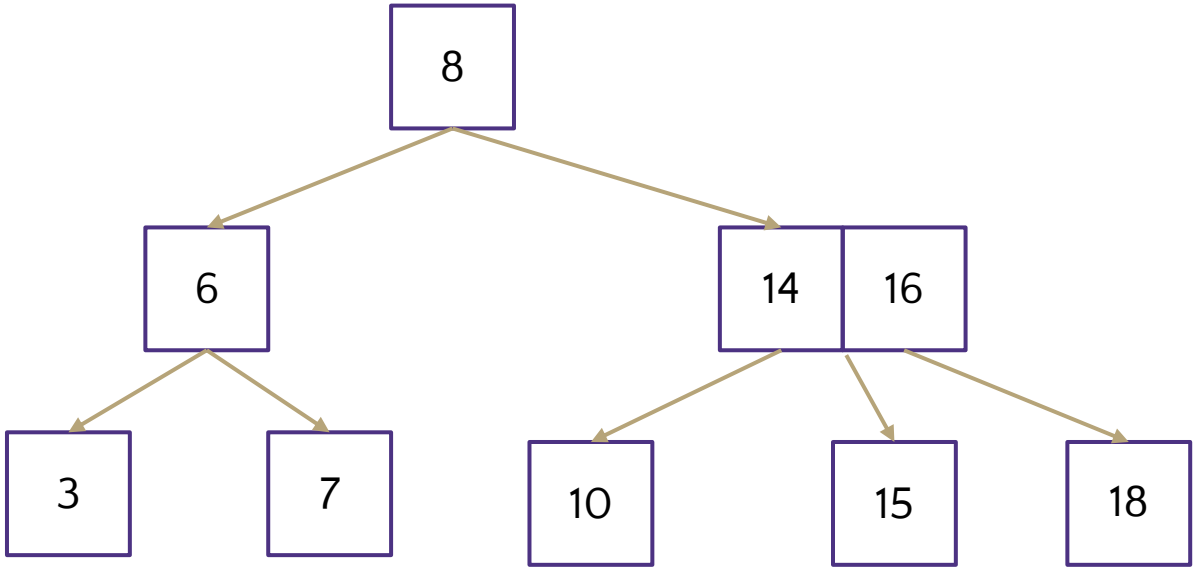
Final Exam Sample Questions Spring 2025 ANS

Department of Computer Science,

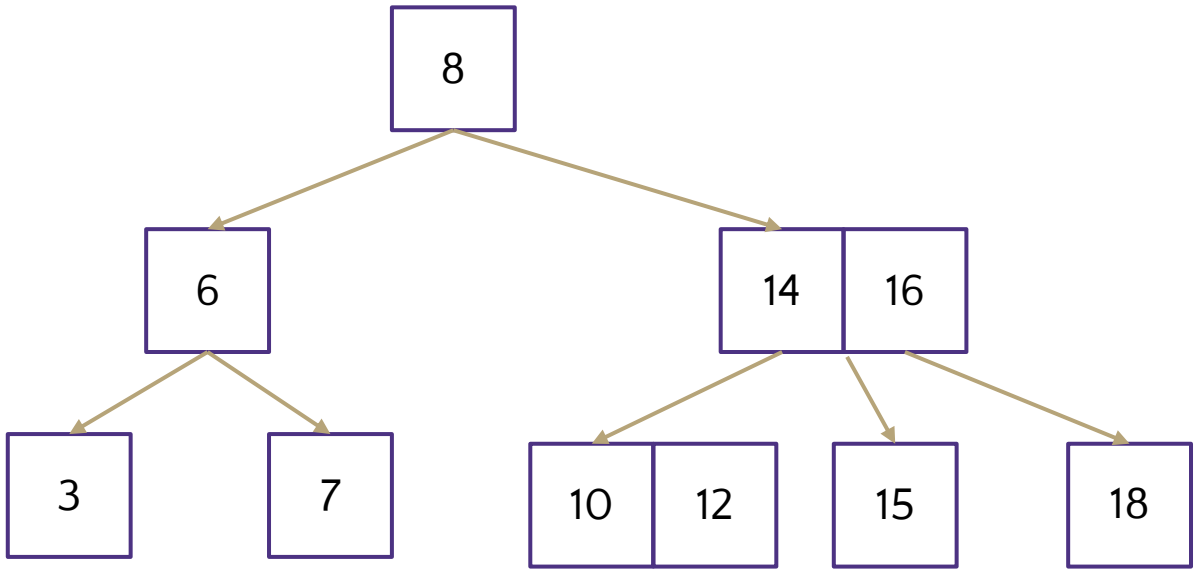
Hofstra University

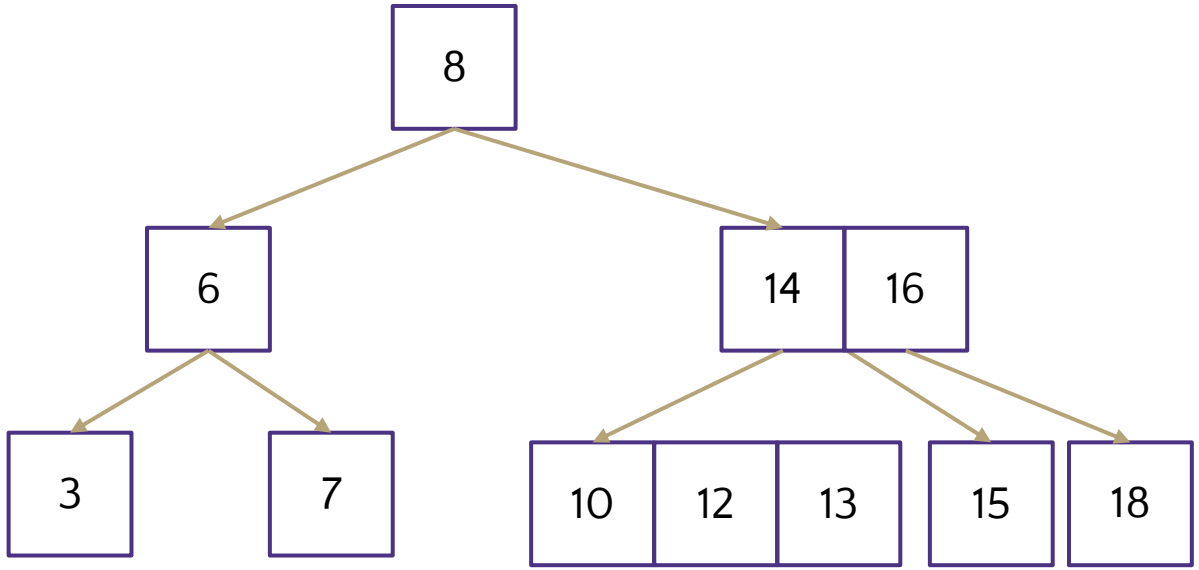
L10 2-3 Trees

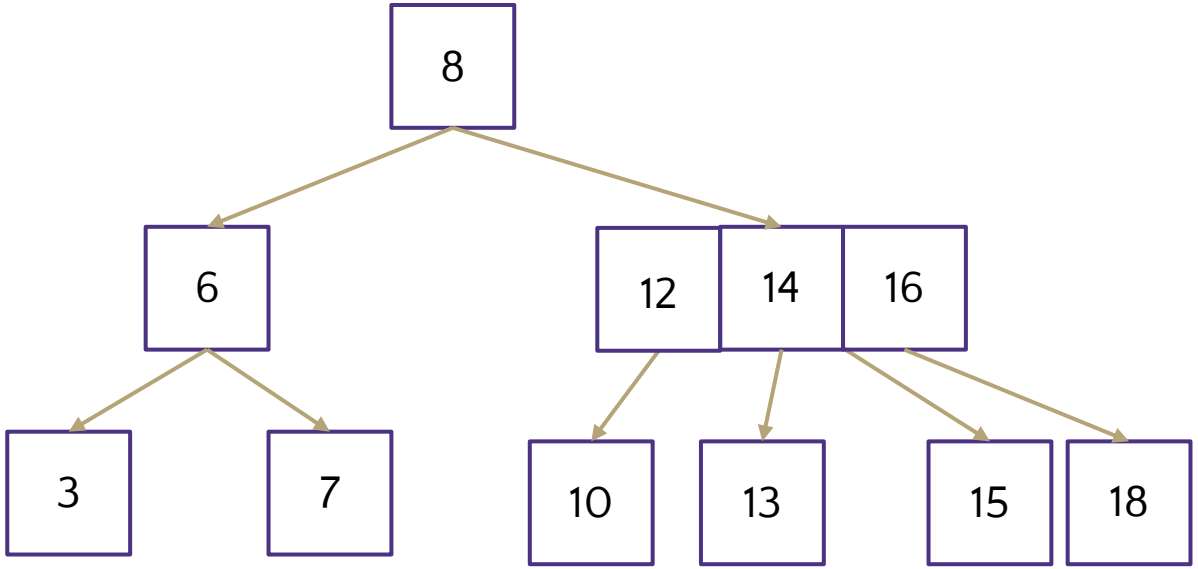
- Insert keys 12 and 13 into the following 2-3 tree. Show the detailed steps after inserting each item

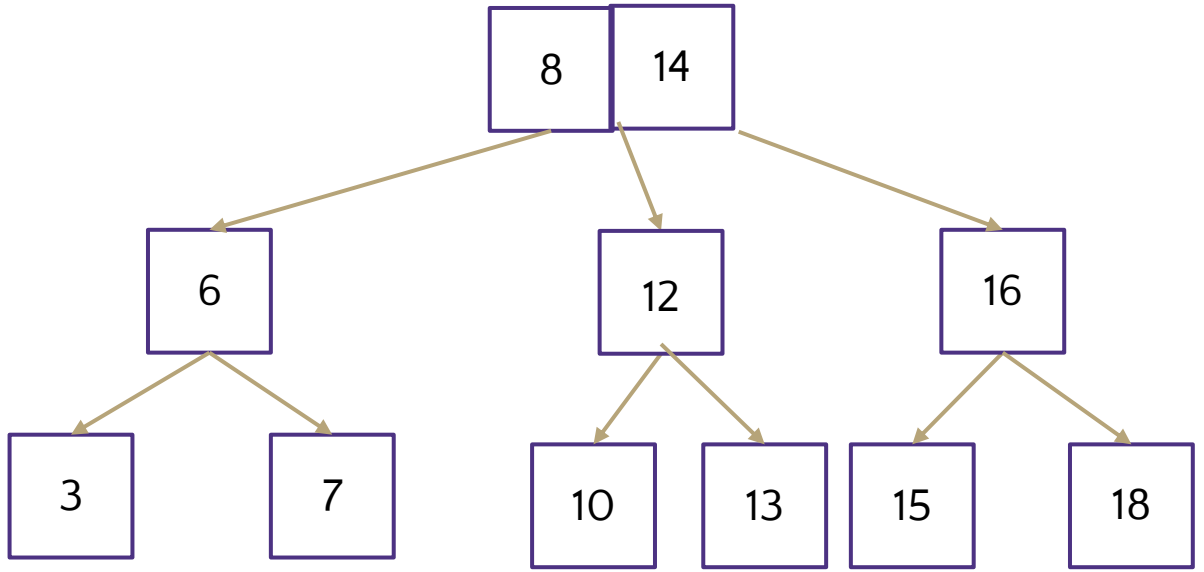


L10 2-3 Trees ANS







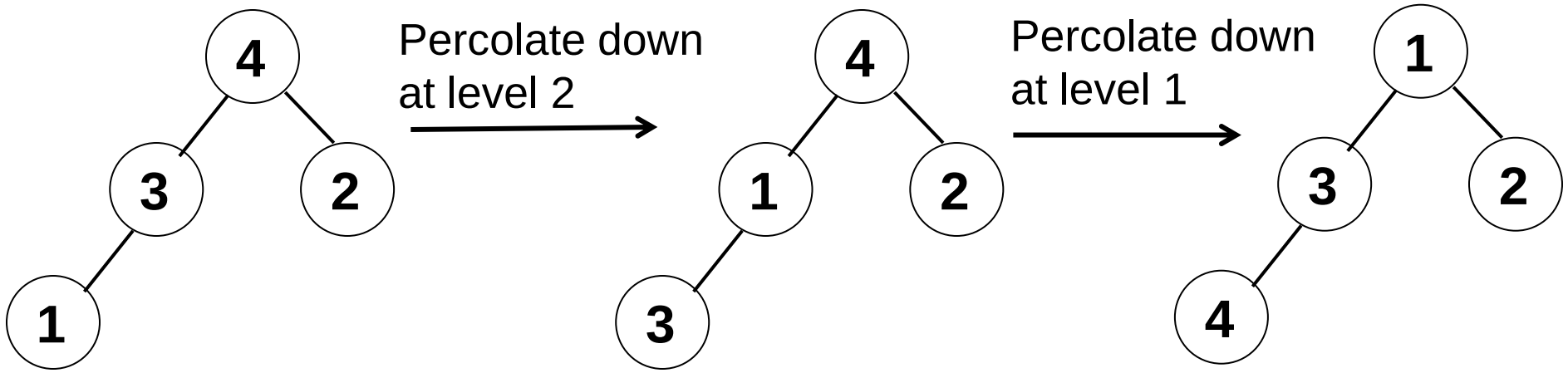


L11 Heaps

- Consider the following sequence of numbers: 4, 3, 2, 1. Build a binary min-heap with these numbers in two ways.
- (a) Use Floyd's build-heap to build the heap. Draw the heap before and after each percolation. At the end, draw the array representation of the final heap.
- (b) Build the heap using repeated insertions (in the order given: 4, 3, 2, 1) - draw the heap after each insertion. At the end, draw the array representation of the final heap.

L11 Heaps ANS

- Consider the following sequence of numbers: 4, 3, 2, 1. Build a binary min-heap with these numbers in two ways.
- (a) Use Floyd's build-heap to build the heap. Draw the heap before and after each percolation. At the end, give the array representation of the final heap.

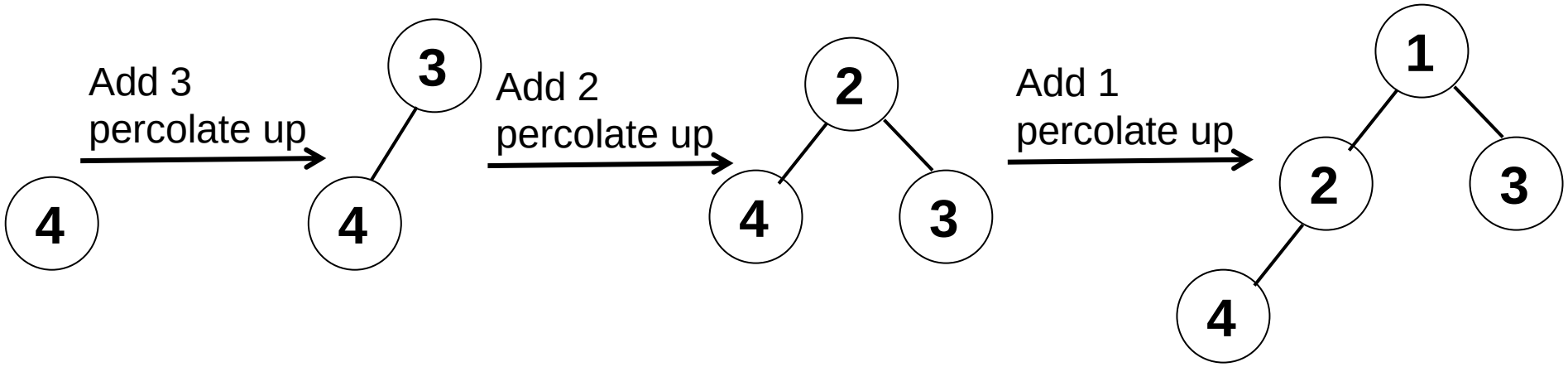


Array representation

1	3	2	4

L11 Heaps ANS

- Consider the following sequence of numbers: 4, 3, 2, 1. Build a binary min-heap with these numbers in two ways.
- (b) Build the heap using repeated insertions (in the order given: 4, 3, 2, 1) - draw the heap after each insertion. At the end, draw the array representation of the final heap.

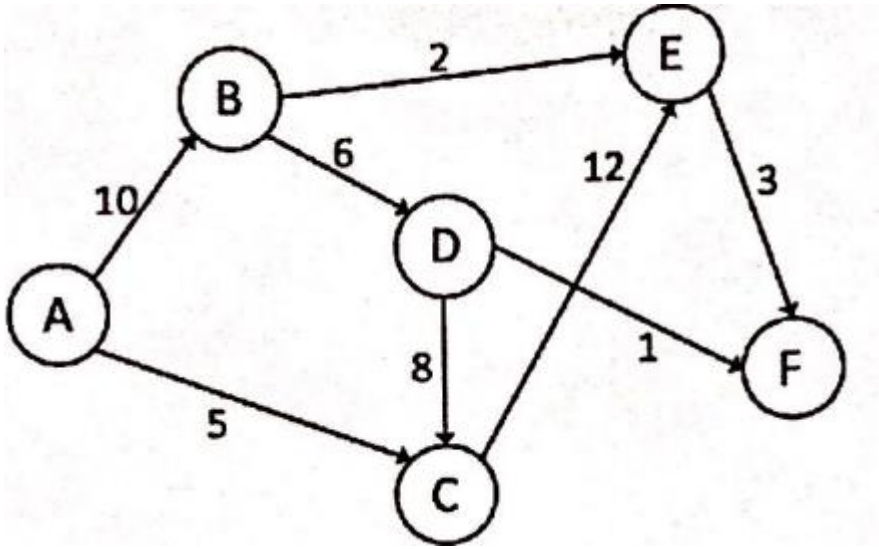


Array representation

1	2	3	4

L13 Dijkstra's Algorithm

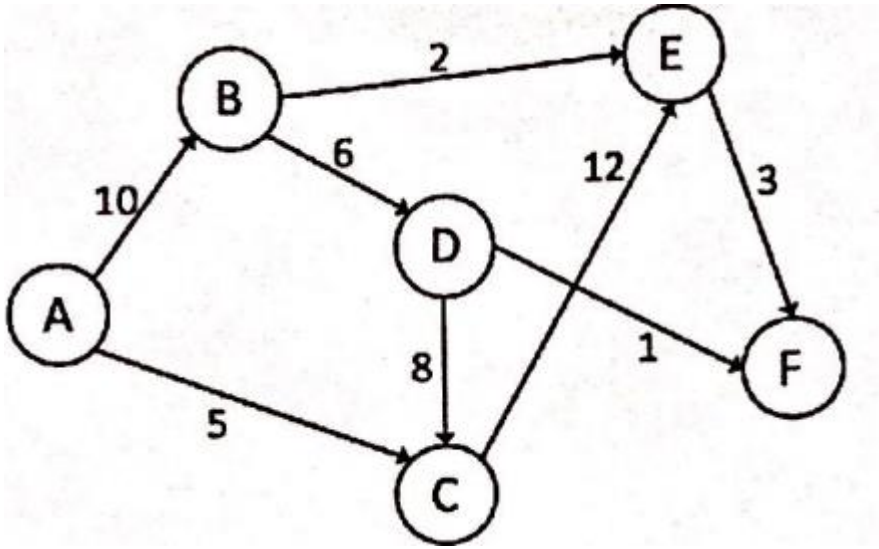
- Given this directed graph, run Dijkstra's Algo to find shortest paths starting from source node A. Give the node visit order, and fill in this table of SN (Shortest Distance) and PN (Previous Node), crossing out old SD and PN as you find a shortcut path with smaller SD.



Node	SD	PN
A		
B		
C		
D		
E		
F		

L13 Dijkstra's Algorithm ANS

- Given this directed graph, run Dijkstra's Algo to find shortest paths starting from source node A. Give the node visit order, and fill in this table of SN (Shortest Distance) and PN (Previous Node), crossing out old SD and PN as you find a shortcut path with smaller SD.



Node	SD	PN
A	0	/
B	10	A
C	5	A
D	16	B
E	~~17~~ 12	~~C~~ B
F	15	E

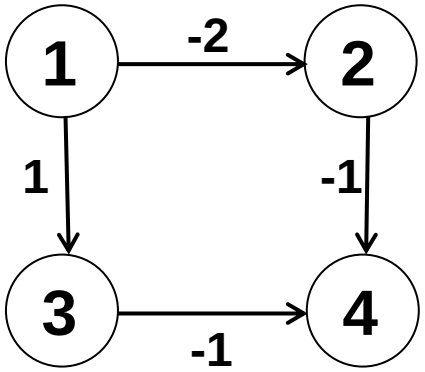
L13 Johnson's algorithm

Consider the following weighted digraph. As part of Johnson's algorithm for All-pairs Shortest Paths, add a dummy source node d, and edges with weight 0 from d to all vertices of G. Let the modified graph be G'.

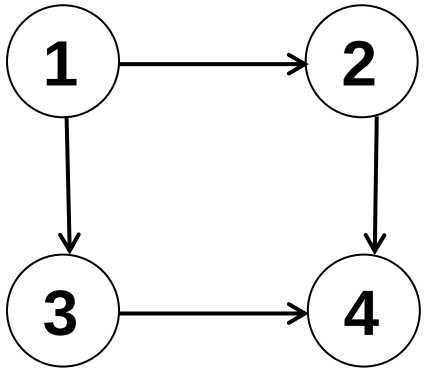
a) Compute the shortest distances from dummy source node d to each node in G' by hand: h[0], h[1], .. h[V-1], then reweight the edges of the original graph to make the edge weights greater than or equal to 0. Draw the reweighted graph G' (without the dummy node d).

b) For the reweighted graph G': run Dijkstra's Algo to find shortest paths starting from source node 1, and compute the shortest paths for the graph with updated positive or zero weights. (Do not show the intermediate steps.)

c) For the original graph G: compute the shortest paths starting from source node 1 with negative weights.



Original graph



Reweighted graph

Node	h()
1	
2	
3	
4	

Shortest paths starting from dummy node

Node	SD	PN
1	0	/
2		
3		
4		

Shortest paths starting from source node 1 in reweighted graph

Node	SD	PN
1	0	/
2		
3		
4		

Shortest paths starting from source node 1 in original graph

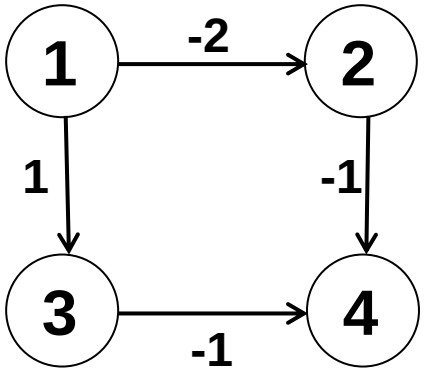
Q. Johnson's algorithm ANS (a)(b)

Shortest distances from dummy source node d: h[1]=0, h[2]=-2, h[3]=0, h[4]=-1. (Theoretically you should run Bellman-Ford algorithm starting from node d, but the graph is simple enough that you can obtain the h[] values by observation.)

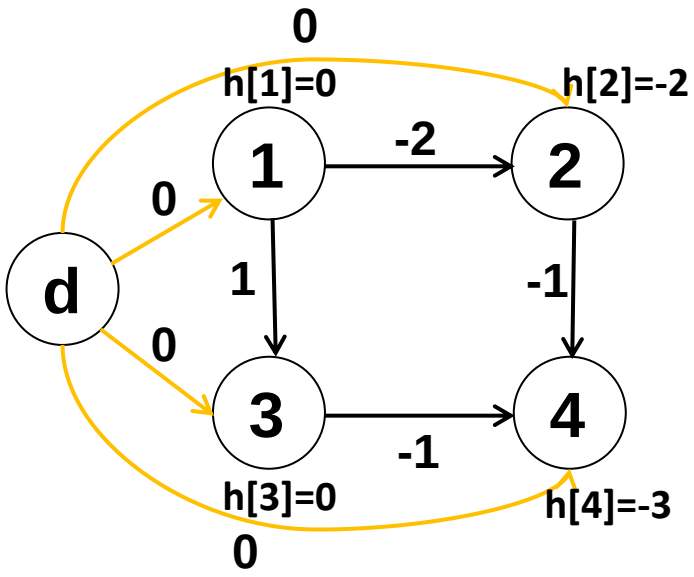
Remove node d and reweight each edge uv as w'(u, v) = w(u, v) + h[u] – h[v], we have: w'[1, 2]=-2+0-(-2)=0, w'[1, 3]=1+0-0=1, w'[2, 4]=-1+(-2)-(-3)=0, w'[3, 4]=-1+0-(-3)=2

Node	h()
1	0
2	-2
3	0
4	-3

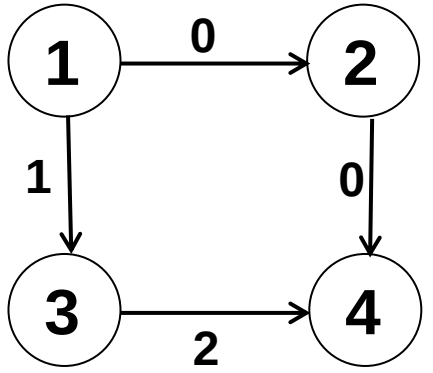
Shortest paths starting from dummy node



Original graph



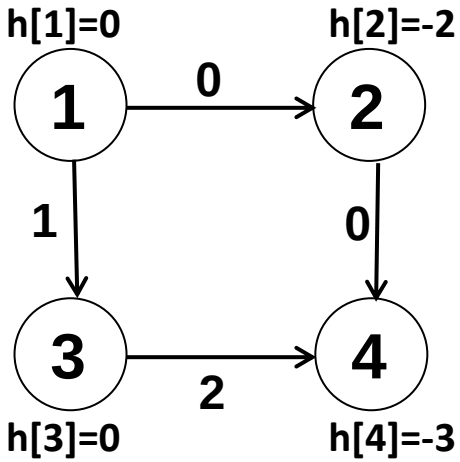
Original graph w/ dummy node



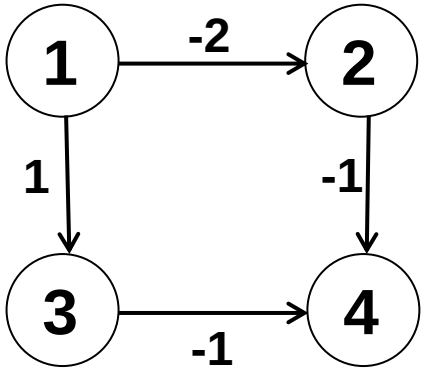
Reweighted graph

Q. Johnson's algorithm ANS (c)

- Let's run Dijkstra's algorithm starting from source node 1, and obtain the shortest paths table for the reweighted graph
- We then subtract h[s] – h[t] from length of each shortest path from s to t to obtain the shortest paths table for the original graph (PN stays the same)
- SD(2)=SD'(2)-(h(1)-h(2))=0–(0-(-2))=-2
- SD(3)=SD'(3)-(h(1)-h(3))=1-(0-0)=1
- SD(4)=SD'(4)-(h(1)-h(4))=0-(0-(-3))=-3



Reweighted graph



Original graph

Node	h()
1	0
2	-2
3	0
4	-3

Shortest paths starting from dummy node

Node	SD'	PN
1	0	/
2	0	1
3	1	1
4	0	2

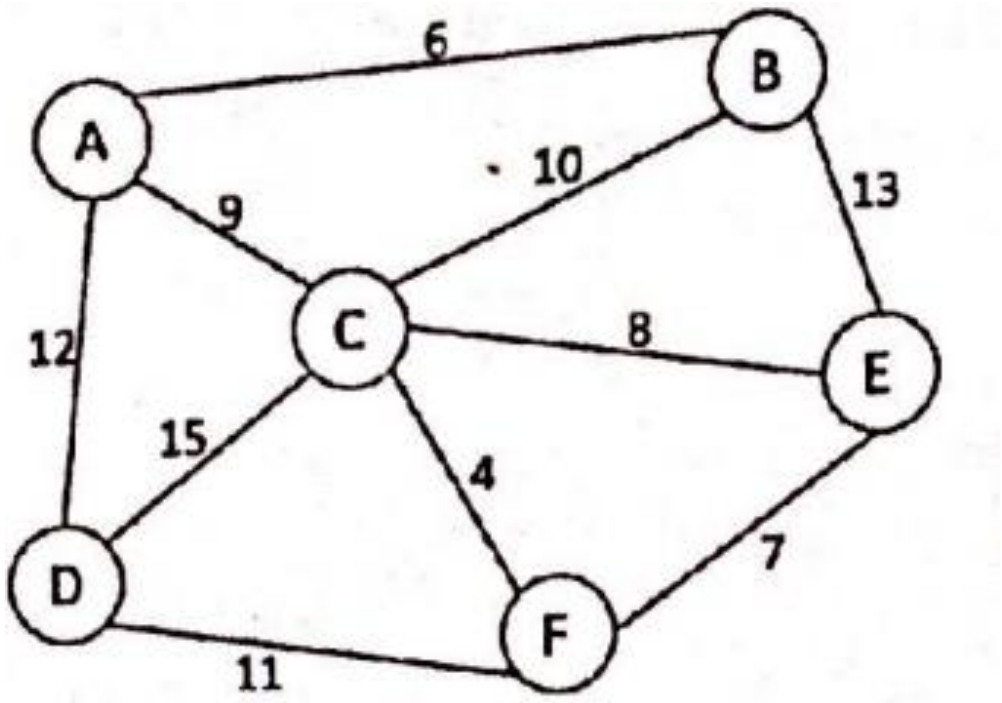
Shortest paths starting from source node 1 in reweighted graph

Node	SD	PN
1	0	/
2	-2	1
3	1	1
4	-3	2

Shortest paths starting from source node 1 in original graph

L15 MST Prim's

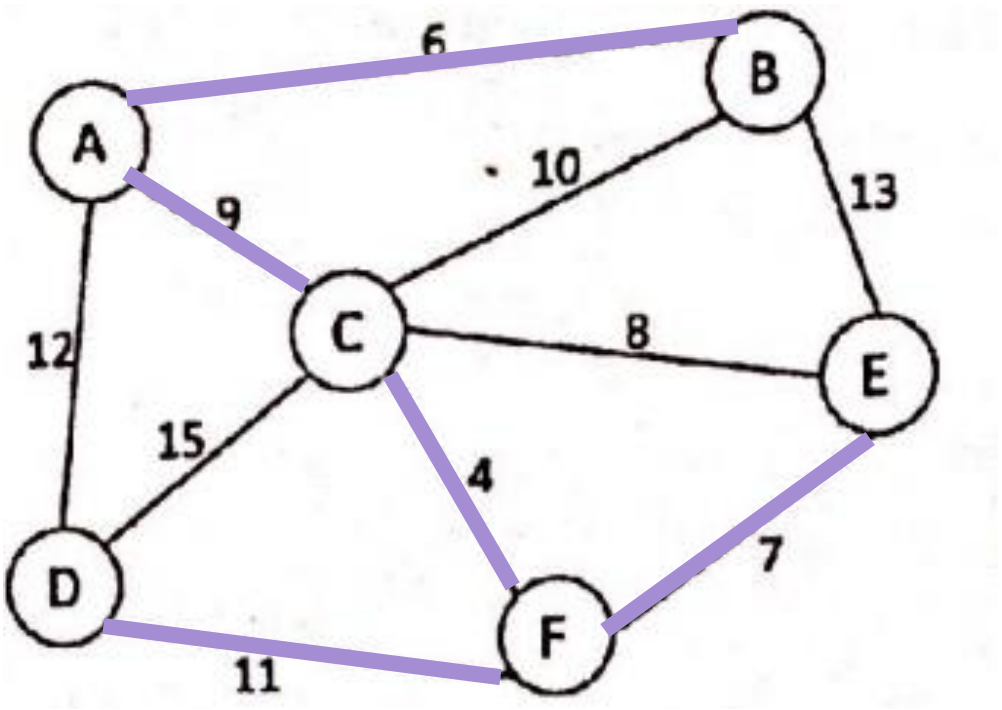
- Run Prim's algorithm starting from node A. Fill in the table with the order in which each edge is added, and its weight. Break ties in alphabetical or numerical order. Draw the final MST. For an undirected edge, write the nodes in alphabetical order, e.g., (E, F) instead of (F, E).



Order added	Edge	Edge Weight
1		
2		
3		
4		
5		

L15 MST Prim's ANS

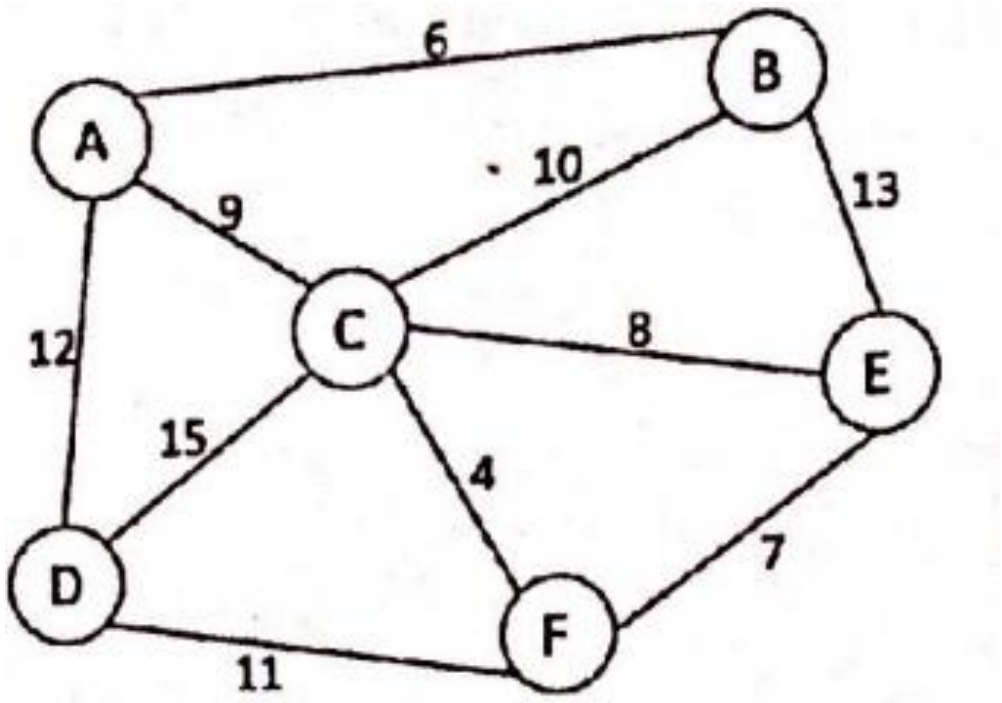
- Run Prim's algorithm starting from node A. Fill in the table with the order in which each edge is added, and its weight. Break ties in alphabetical or numerical order. Draw the final MST. For an undirected edge, write the nodes in alphabetical order, e.g., (E, F) instead of (F, E).



Order added	Edge	Edge Weight
1	(A, B)	6
2	(A, C)	9
3	(C, F)	4
4	(E, F)	7
5	(D, F)	11

L15 MST Kruskal's

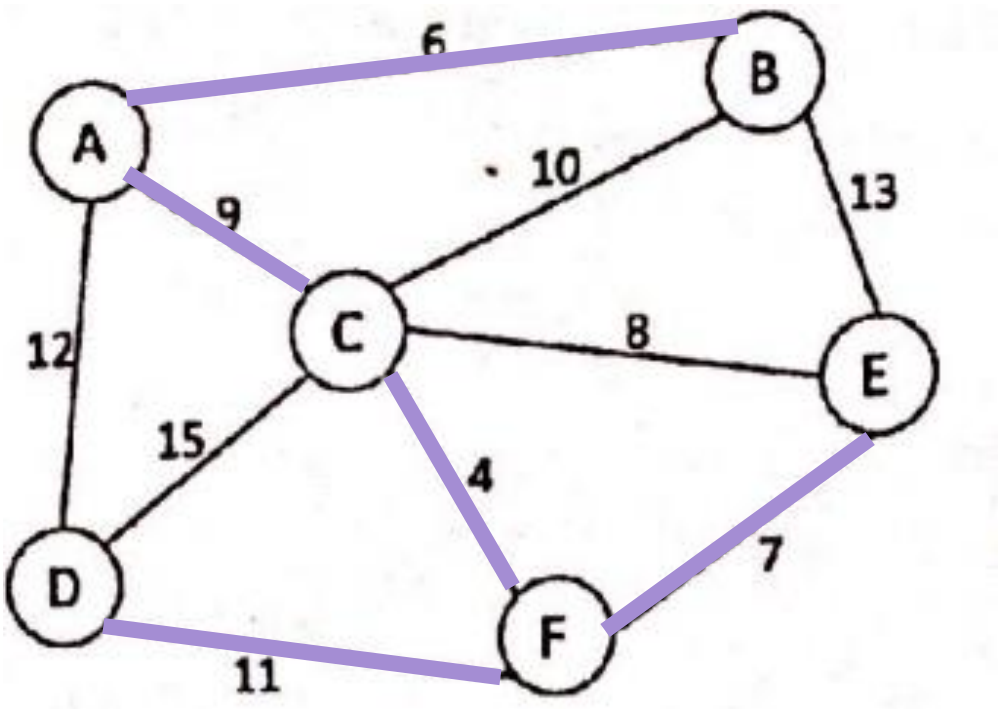
- Run Kruskal's algorithm. Fill in the table with the order in which each edge is added, and its weight. Break ties in alphabetical or numerical order. Draw the final MST. For an undirected edge, write the nodes in alphabetical order, e.g., (E, F) instead of (F, E).



Order added	Edge	Edge Weight
1		
2		
3		
4		
5		

L15 MST Kruskal's ANS

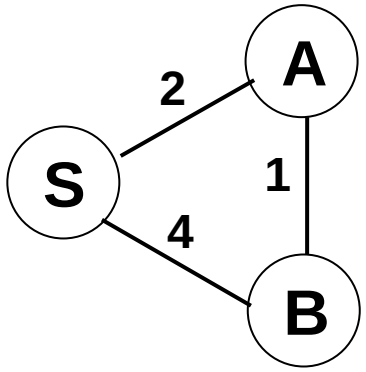
- Run Kruskal's algorithm. Fill in the table with the order in which each edge is added, and its weight. Break ties in alphabetical or numerical order. Draw the final MST. For an undirected edge, write the nodes in alphabetical order, e.g., (E, F) instead of (F, E).



Order added	Edge	Edge Weight
1	(C, F)	4
2	(A, B)	6
3	(E, F)	7
4	(A, C)	9
5	(D, F)	11

L15 MST

- (a) Run Prim's algorithm starting from node S to find the MST. Fill in the table with the order in which each edge is added, and its weight. Break ties in alphabetical or numerical order. Draw the final MST. For an undirected edge, write the nodes in alphabetical order, e.g., (E, F) instead of (F, E).
- (b) Run Kruskal's algorithm to find the MST.
- (c) Run Dijkstra's algorithm starting from node S to find the shortest paths from node S. Draw the shortest path tree.



Order added	Edge	Edge Weight
1		
2		

Prim's algorithm

Order added	Edge	Edge Weight
1		
2		

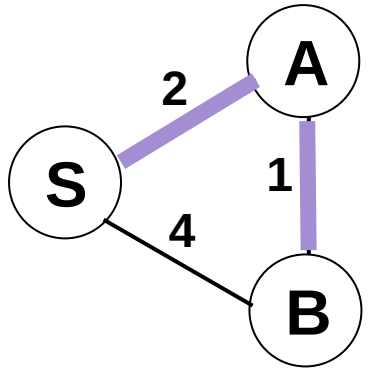
Kruskal's algorithm

Node	SD	PN
S	0	/
A		
B		

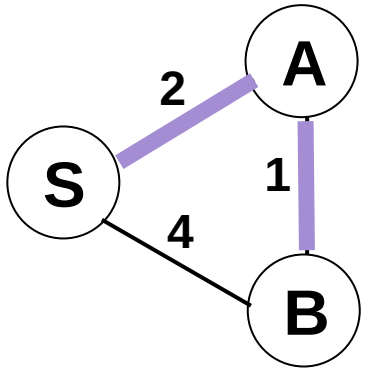
Dijkstra's algorithm

L15 MST ANS

- (a) Run Prim's algorithm starting from node S to find the MST. Fill in the table with the order in which each edge is added, and its weight. Break ties in alphabetical or numerical order. Draw the final MST. For an undirected edge, write the nodes in alphabetical order, e.g., (E, F) instead of (F, E).
- (b) Run Kruskal's algorithm to find the MST.
- (c) Run Dijkstra's algorithm starting from node S to find the shortest paths from node S. Draw the shortest path tree.



Shortest path tree from S



MST

Order added	Edge	Edge Weight
1	(A, S)	2
2	(A, B)	1

Prim's algorithm

Order added	Edge	Edge Weight
1	(A, B)	1
2	(A, S)	2

Kruskal's algorithm

Node	SD	PN
S	0	/
A	2	S
B	4 3	~~S~~ A

Dijkstra's algorithm
Visit order: S, A, B