**CSC017 Fall 2024 Final Exam**

**Student Name： ID：**

|  |  |
| --- | --- |
| **Total Points** |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **5** | **6** | **6** | **4** | **4** | **5** | **5** | **5** | **6** | **6** |
| 11 | 12 | 13 | 14 |  |  |  |  |  |  |
| **6** | **4** | **5** | **5** |  |  |  |  |  |  |

1. (5 pts) Lecture 5-algorithm performance analysis

For each function *f*(*n*) below, give an asymptotic upper bound using big-O notation. You should give the tightest bound possible (so giving *O*(2*n*) for every question is unlikely to result in many points).

1. *f*(*n*) = 100*n*3 − 7*n*3 + 14*n*2
2. *f*(*n*) = 100*n*3 − 100*n*3 + 7*n*2
3. *f*(*n*) = log(7*n*2)
4. *f*(*n*) = 5loglog*n* + 4log2(*n*)
5. *f*(*n*) = *.*001*n* + 100 · 2*n*
6. *f*(*n*) = *n*3(1 + 6*n* + 2014*n*2)
7. *f*(*n*) = (log*n*)(*n* + *n*2)

**ANS:**

1. *O*(*n*3)
2. *O*(*n*2)
3. *O*(log*n*)
4. *O*(log2 *N*)
5. *O*(2*n*)
6. *O*(*n*5)

(g) *O*(*n*2 log*n*)

2. (6 pts) Lecture 7-hash table

Insert the following six keys in this order: 19, 48, 8, 27, 97, 7 into a hash table of size 10, where the hash function is modulo table size (%10).

a) (1 pts) What is the load factor?

ANS: Load factor = number of keys / size of hash table = 6/10 = 0.6

b) (2 pts) Fill in the table, resolving hash collisions with linear probing.

ANS:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 97 | 7 |  |  |  |  | 27 | 48 | 19 |

c) (2 pts) Fill in the table, resolving hash collisions with quadratic probing.

ANS:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | 97 | 8 |  |  |  | 7 | 27 | 48 | 19 |

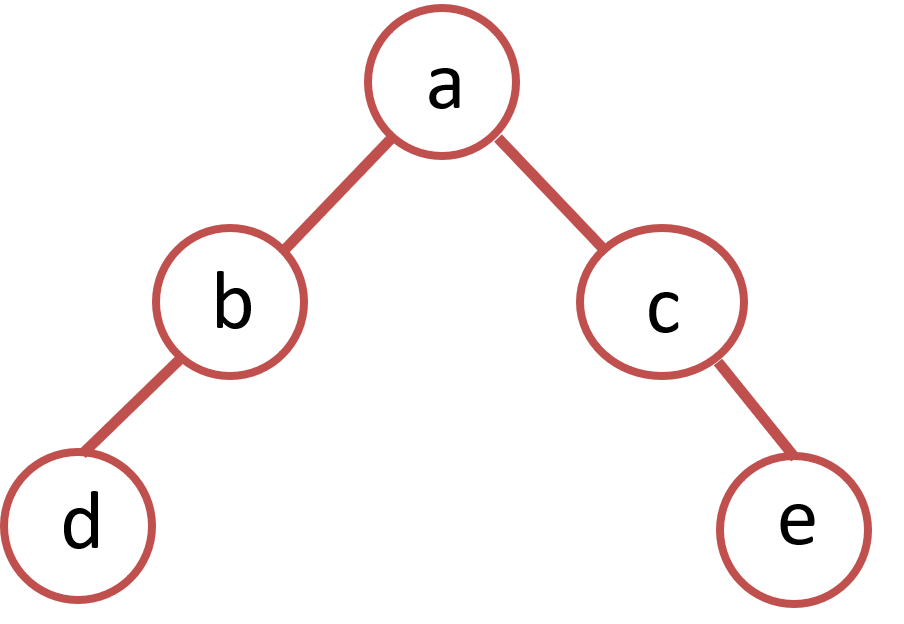
d) (1 pts) Fill in the table, resolving hash collisions with separate chaining into a sorted linked list (with the smallest element at the head of the list).

ANS:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  |  |  |  |  | 7,27,97 | 8,48 | 19 |

3. (6 pts) Lecture 8-binary search tree and trie

a) (2 pts) Give Preorder, In-order, Post-order (by DFS), and Level-order traversal (by BFS) of this tree:



ANS: Preorder: abdce

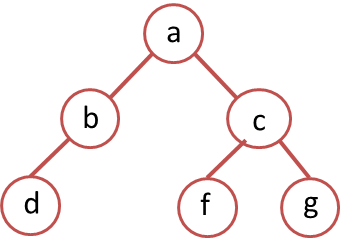
In-order: dbace

Post-order: dbaec

Level-order: abcde

b) (4 pts) The in-order and pre-order traversal of a binary tree are: abdcfg and dbafcg, respectively. a) Draw this binary tree. b) Give the post-order traversal of this binary tree.

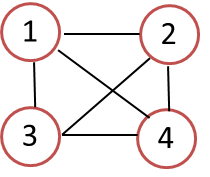
ANS:



Post-order: dbfgca

4. (4 pts) Lecture 10-basic graph algorithms

a) (2 pts) Write out the *adjacency list* and *adjacency matrix* representations of the following *undirected* graph. b) (2 pts) Write out the *adjacency matrix* representation of 2-hop neighbors of the following *undirected* graph.



a) Adjacency list:

1 → {\_\_\_\_\_\_\_}

2 → {\_\_\_\_\_\_\_}

3 → {\_\_\_\_\_\_\_}

4 → {\_\_\_\_\_\_\_}

Adjacency matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

Adjacency matrix of 2-hop neighbors:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

ANS: Adjacency list:

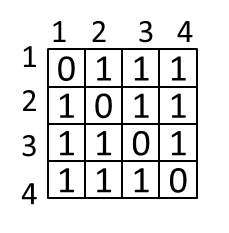
1 → {2, 3, 4}

2 → {1, 3, 4}

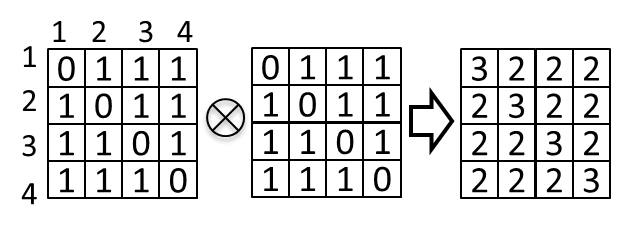
3 → {1, 2, 4}

4 → {1, 2, 3}

Adjacency matrix:



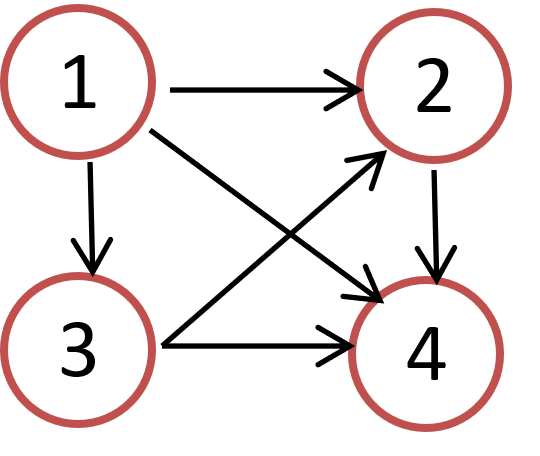
Adjacency matrix of 2-hop neighbors:



5. (4 pts) Lecture 10-basic graph algorithms

a) (2 pts) Write out the *adjacency list* and *adjacency matrix* representations of the following *directed* graph.

b) (2 pts) Write out the *adjacency matrix* representation of 2-hop neighbors of the following *directed* graph.



a) Adjacency list:

1 → {\_\_\_\_\_\_\_}

2 → {\_\_\_\_\_\_\_}

3 → {\_\_\_\_\_\_\_}

4 → {\_\_\_\_\_\_\_}

Adjacency matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

Adjacency matrix of 2-hop neighbors:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

ANS: Adjacency list:

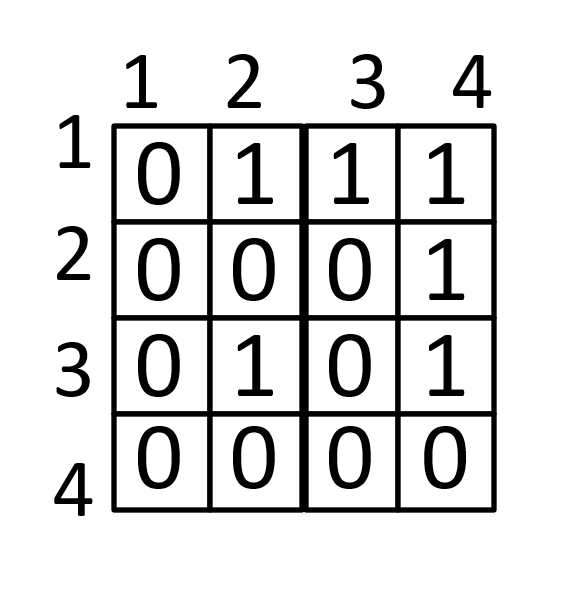
1 → {2, 3, 4}

2 → {1, 3, 4}

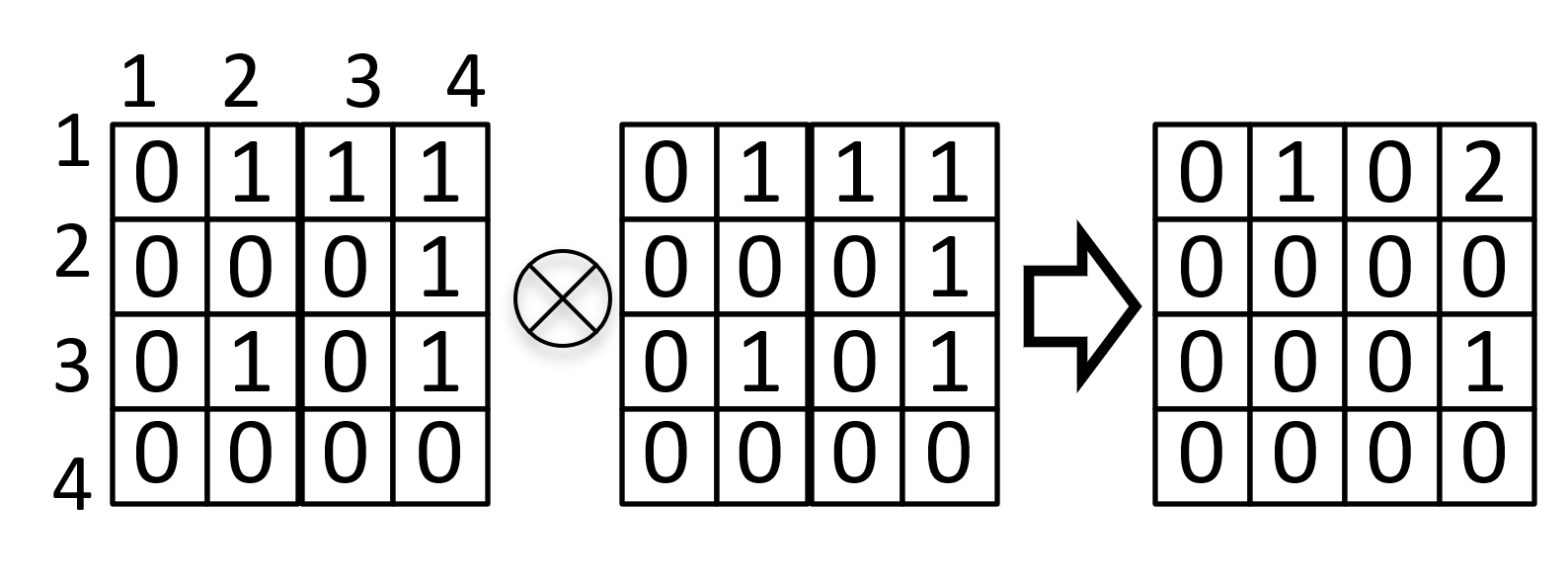
3 → {1, 2, 4}

4 → {1, 2, 3}

Adjacency matrix:



Adjacency matrix of 2-hop neighbors:



6. (5 pts) Lecture 10-basic graph algorithms

Find a topological sort of the following graph, by running depth-first search DFS *starting from node C*, then return vertices in reverse post-order traversal. Show

a) (2 pts) the pre-order traversal;

b) (2 pts) the post-order traversal;

c) (1 pts) the topological sort.

A diagram of a network

Description automatically generated

Pre-order traversal: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Post-order traversal: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Topological sort: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ANS:

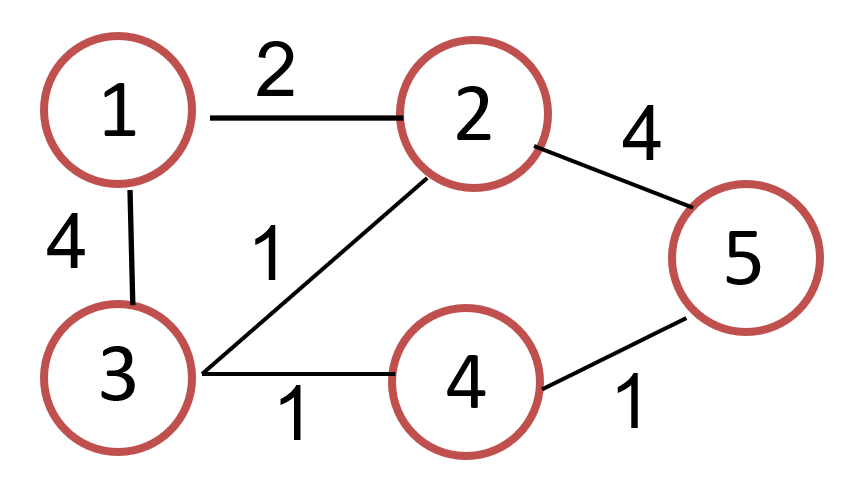
Pre-order traversal: CDEHFGAB

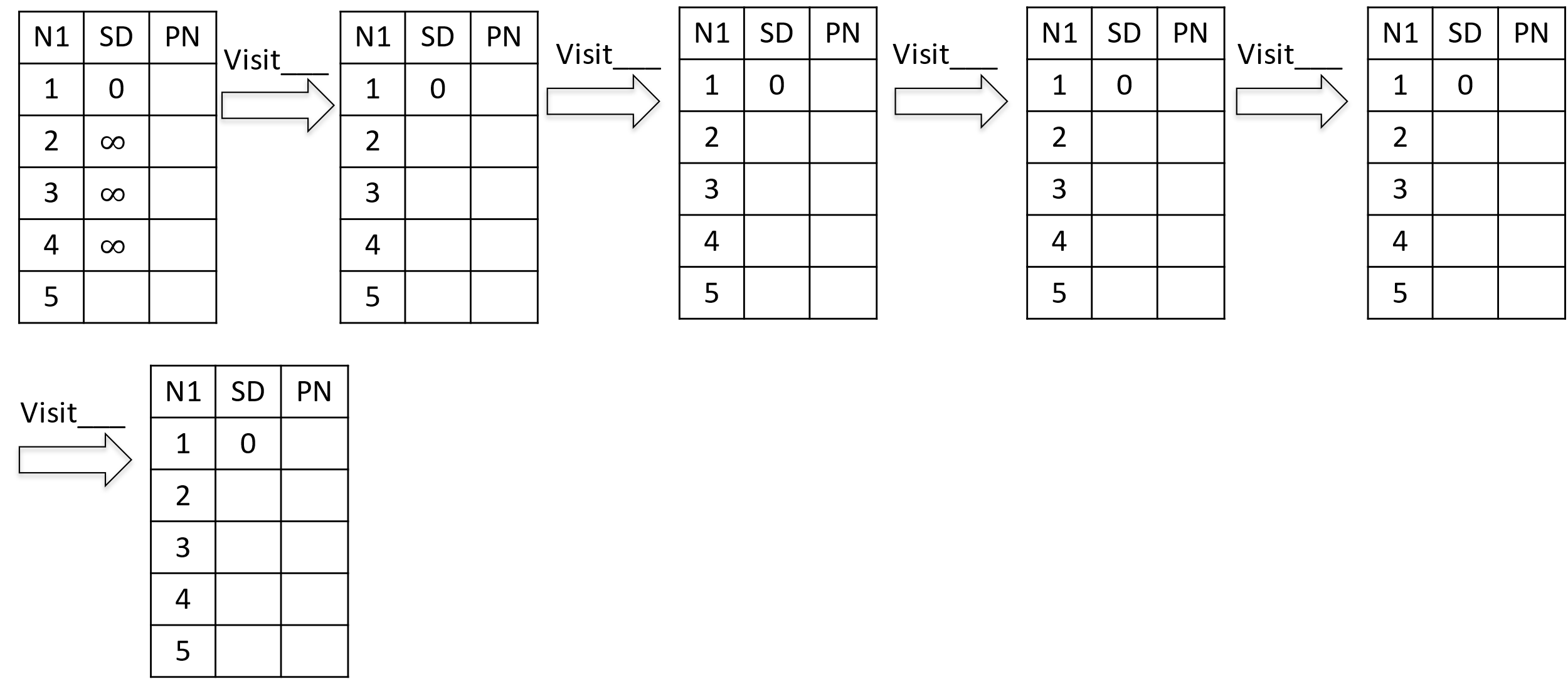
Post-order traversal: HEDGFCBA

Reverse post-order traversal: ABCFGDEH

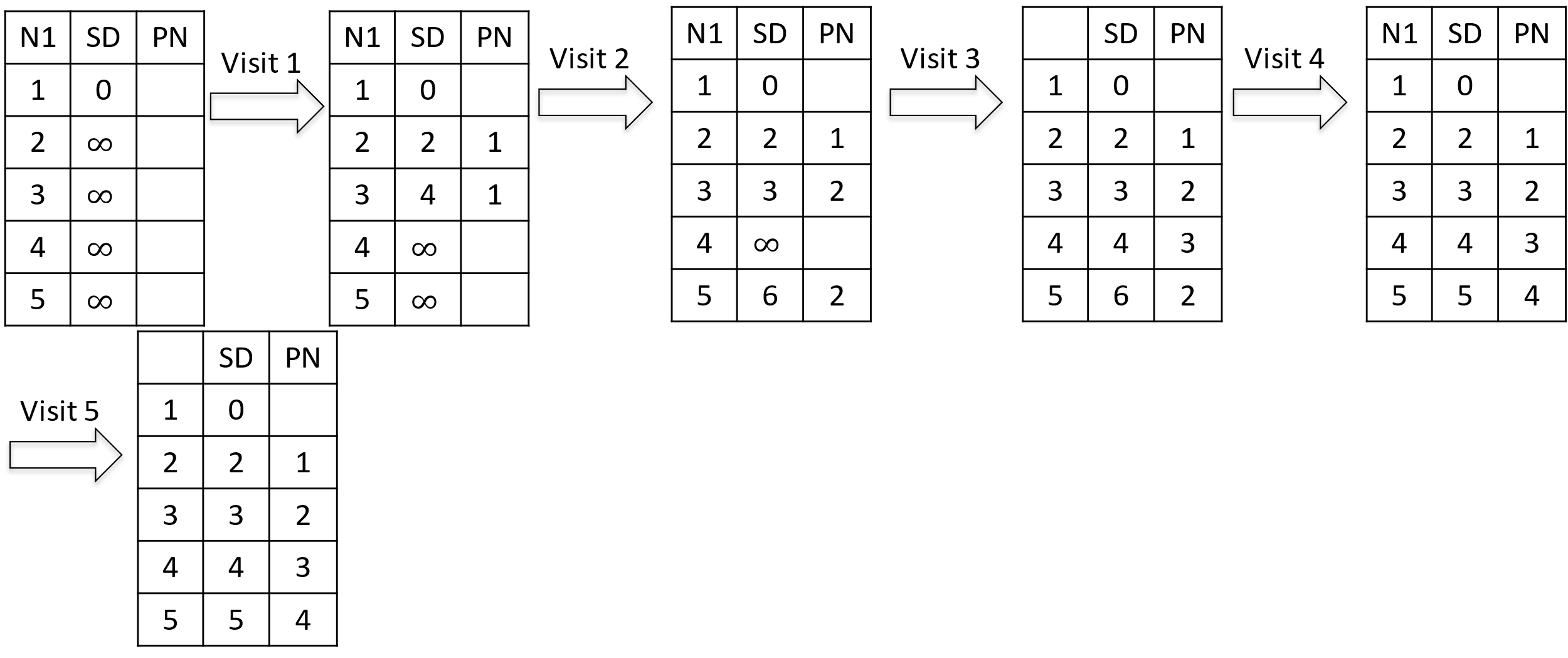
7. (5 pts) Lecture 11-shortest paths

Consider the following undirected graph. Use Dijkstra’s algorithm to find shortest paths starting from source vertex 1. Fill in the tables at each step below, where SD: Shortest Distance. PN: Previous Node.



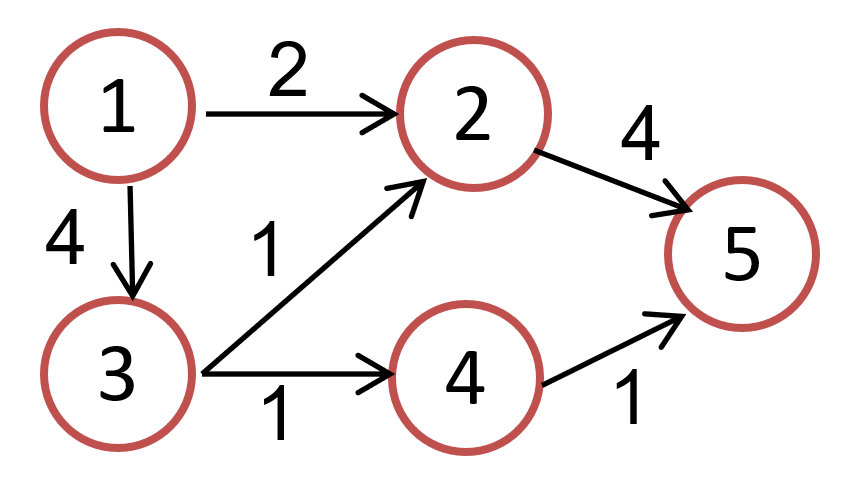


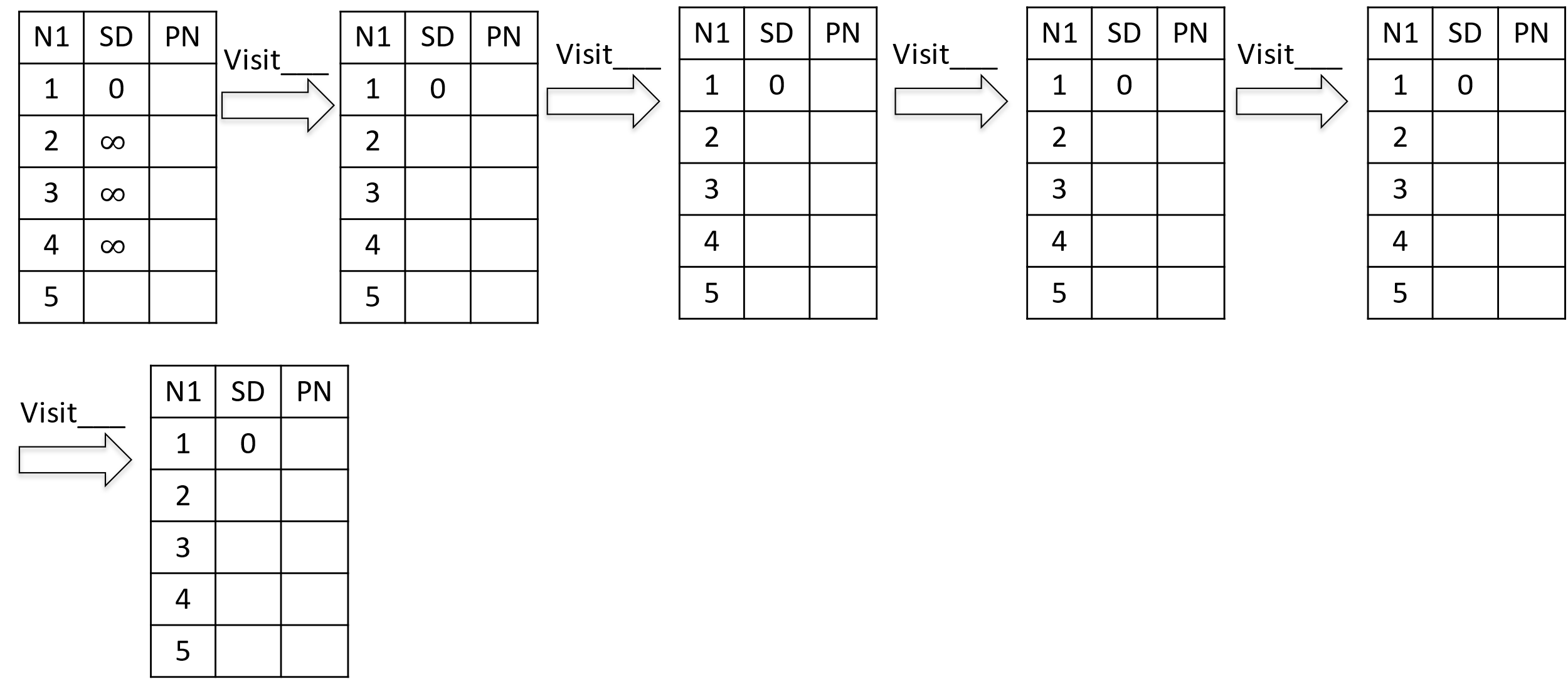
ANS:



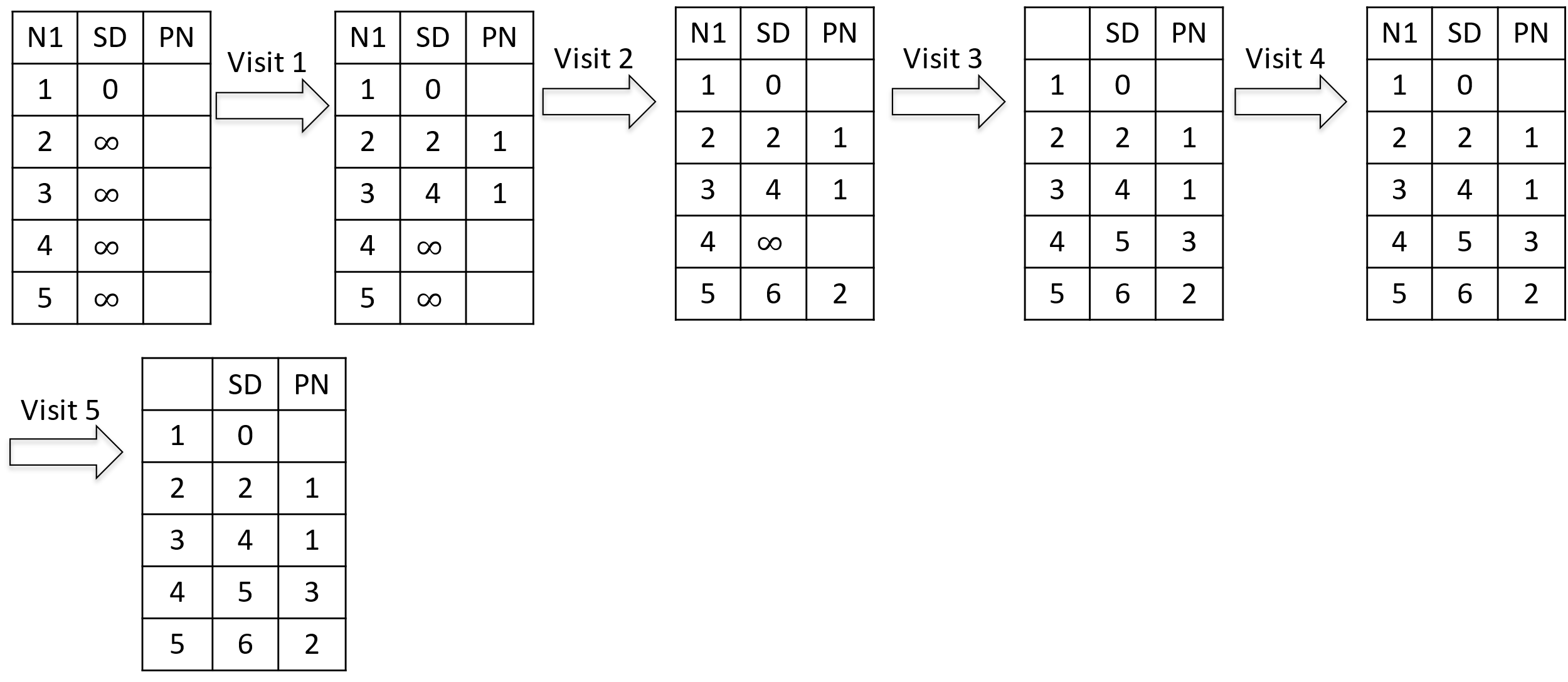
8. (5 pts) Lecture 11-shortest paths

Consider the following directed graph. Use Dijkstra’s algorithm to find shortest paths starting from source vertex 1. Fill in the tables at each step below, where SD: Shortest Distance. PN: Previous Node.



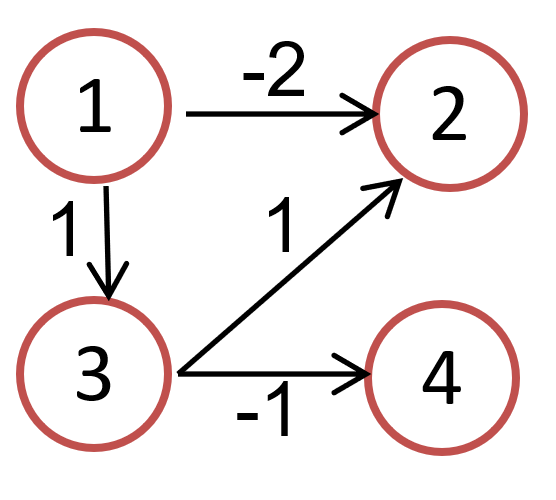


ANS:



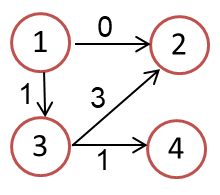
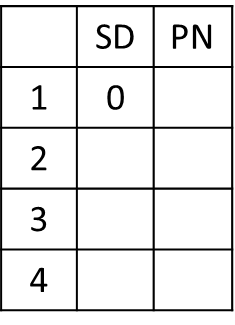
9. (6 pts) Lecture 11-shortest paths

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



a) (2 pts) Compute the shortest distances from dummy source vertex d to each vertex 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.

b) (2 pts) Draw the graph with updated positive or zero weights. Run Dijkstra’s algorithm with source vertex 1, and compute the table of shortest paths for the graph with updated positive or zero weights. (Do not show the intermediate steps.)

c) (2 pts) Compute the table of shortest paths for the original graph G with negative weights.

A black background with a black square

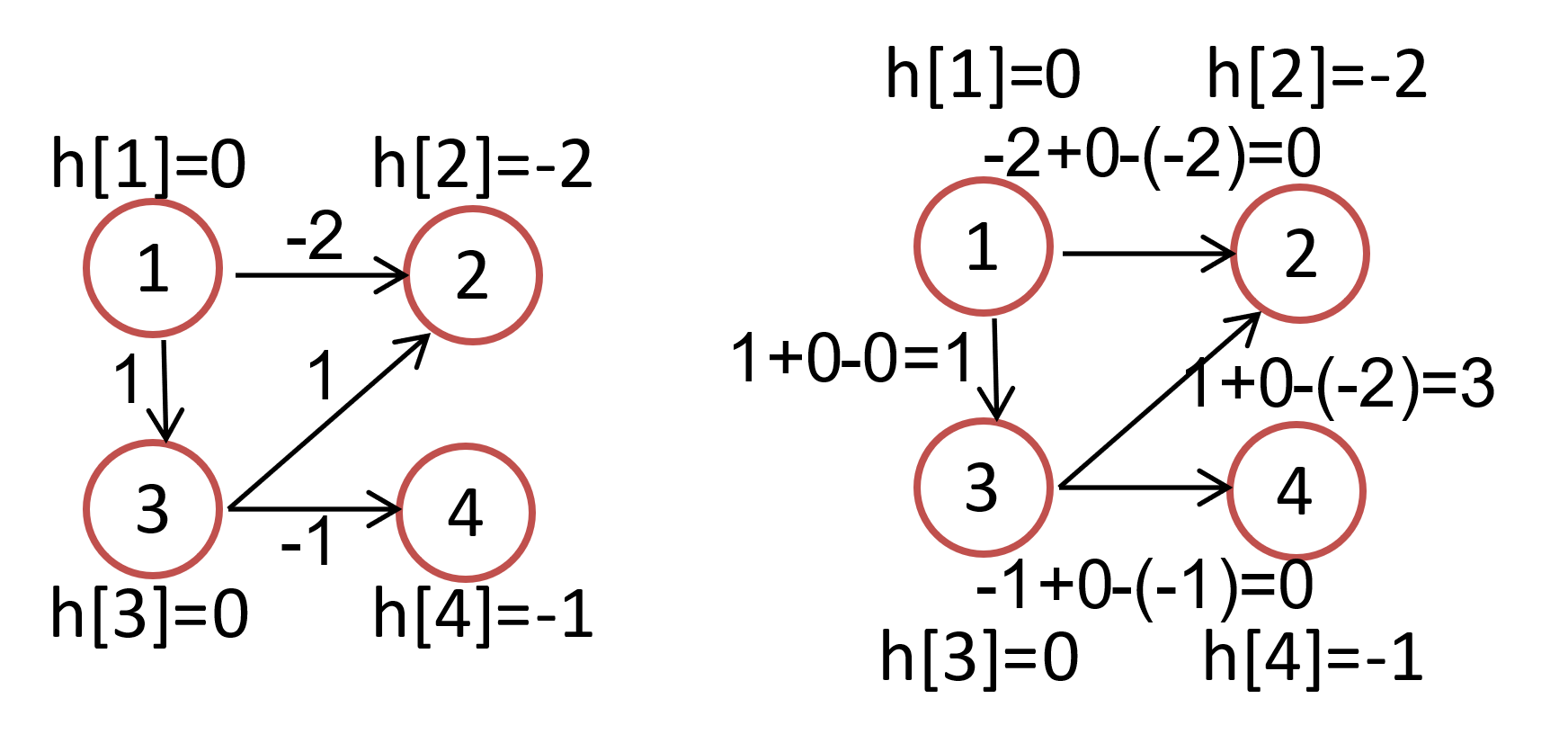
Description automatically generated with medium confidence

ANS:

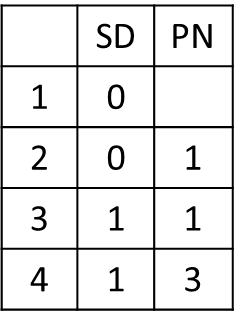
a) h[1]=0, h[2]=-2, h[3]=0, h[4]=-1

w[1][2]=-2+0-(-2)=0, w[1][3]=-1+0-(-1)=0, w[3][2]=1+0-(-2)=3, w[3][4]=-1+0-(-1)=0

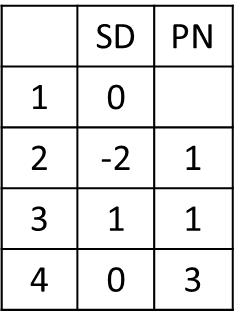
(You do not have to draw the figures below.)



b)

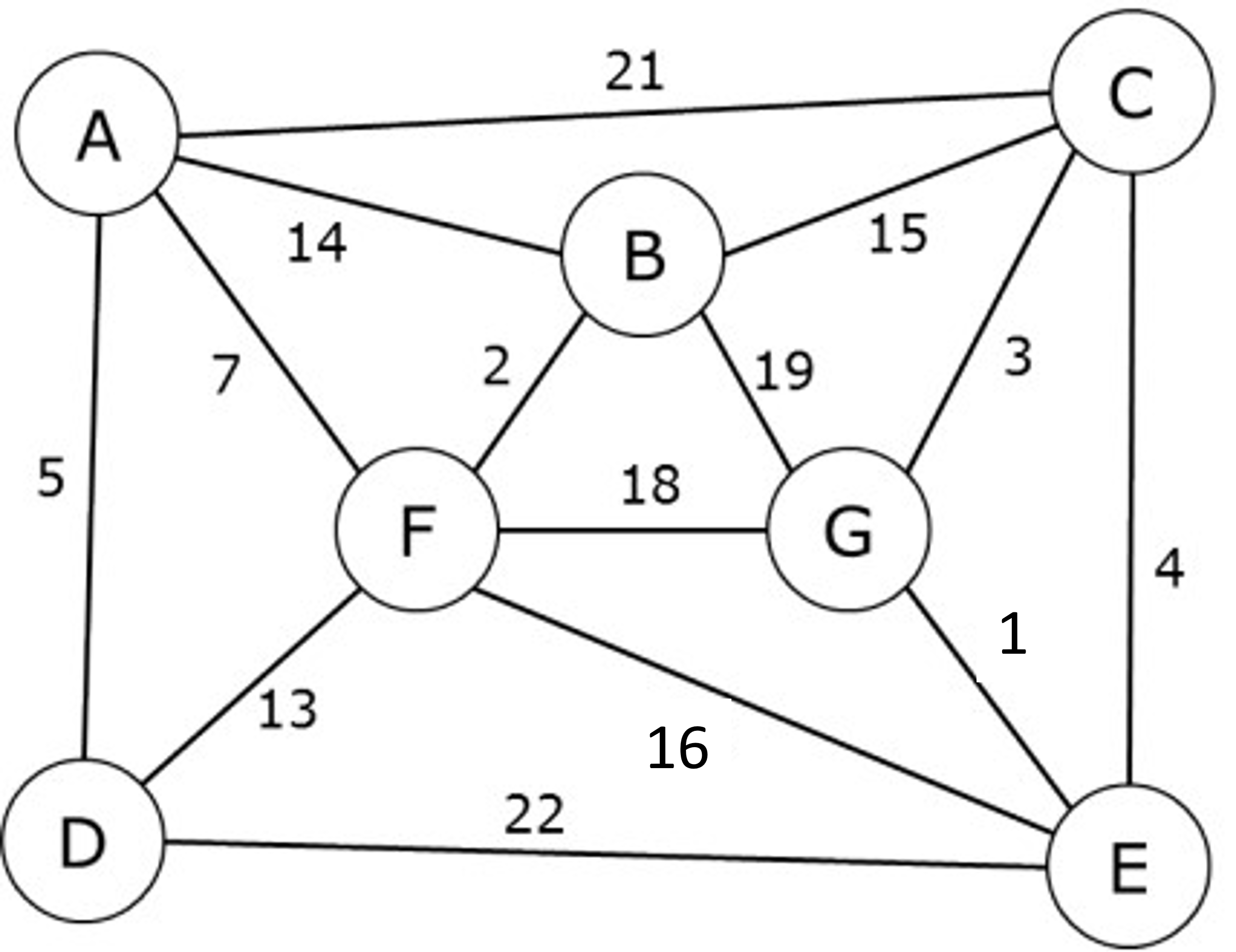


c) SD(2)=0–(0-(-2))=-2, SD(3)=1-(0-0)=1, SD(4)=1 – (0-(-1))=0



10. (6 pts) Lecture 12-minimum spanning trees

a) (3 pts) Use Prim’s algorithm to calculate a minimum spanning tree *starting from vertex A*. If during your algorithm two unvisited vertices have the same distance, use alphabetical order to determine which one is selected first. Name edges in alphabetical order, e.g., write AF instead of FA for the undirected edge. List the edges in the order which Prim’s algorithm includes them into the MST.



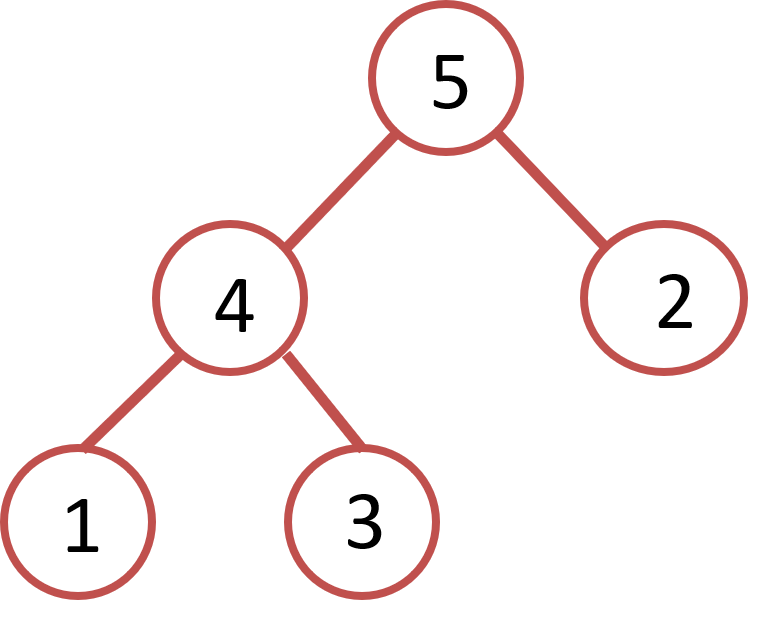
ANS: AD, AF, BF, BC, CG. CE

b) (3 pts) Use Kruskal’s algorithm to calculate a minimum spanning tree. List the edges in the order which Kruskal’s algorithm includes them into the MST.

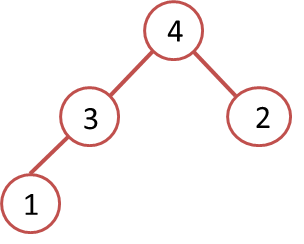
ANS: EG, BF, CG, AD, AF, BC

11. (6 pts) Lecture 13-sorting algorithms

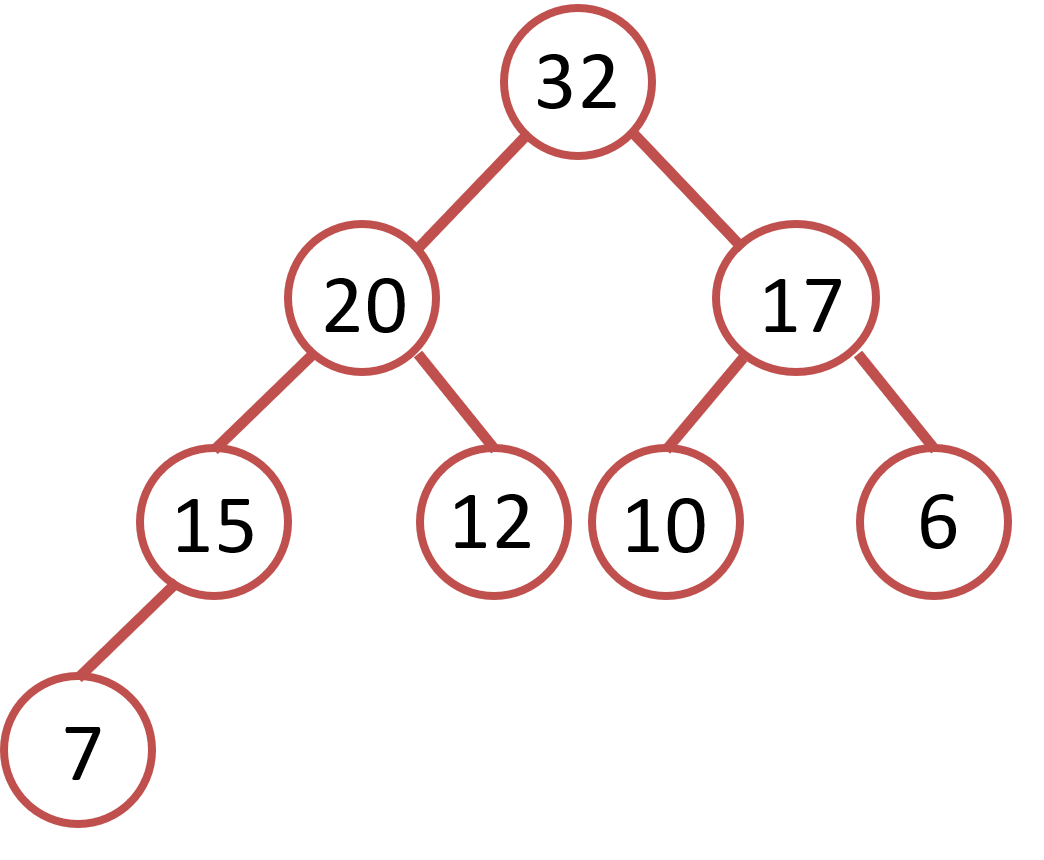
a) (2 pts) Draw the max heap tree constructed by adding each element in this order: 4, 1, 2, 5, 3.



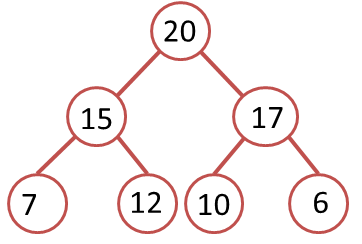
b) (2 pts) Draw the max heap tree after removing the root node from the above-constructed max heap tree.



c) (2 pts) Draw the max heap tree after removing the root node from the below max heap tree.



ANS:



12. (4 pts) Lecture 13-sorting algorithms

Sort this array of numbers with Merge sort into ascending order. Show the split into subarrays, then show the merge steps and give the final sorted array. When there are an odd number of elements in a list, make the left subarray larger. Put an ‘X’ on any empty subarray.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 150 | 41 | 72 | 90 | 802 | 44 | 2 | 686 |

ANS:

150 41 72 90 802 44 2 686

150 41 72 90 802 44 2 686

150 41 72 90 802 44 2 686

41 150 72 90 44 802 2 686

41 72 90 150 2 44 686 802

2 41 44 72 90 150 686 802

13. (5 pts) Lecture 13-sorting algorithms

Sort this array of numbers with Quick sort into ascending order, using *the first number* of each subarray as the pivot. Show the intermediate subarrays at each step, enclosing the pivot at each step with parentheses. Draw the corresponding Binary Search Tree and give the final sorted array.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 150 | 41 | 72 | 90 | 802 | 44 | 2 | 686 |

ANS:

After 1st pass

44 41 72 90 2 (150) 802 686

After 2nd pass, left subarray:

2 41 (44) 90 72

right subarray:

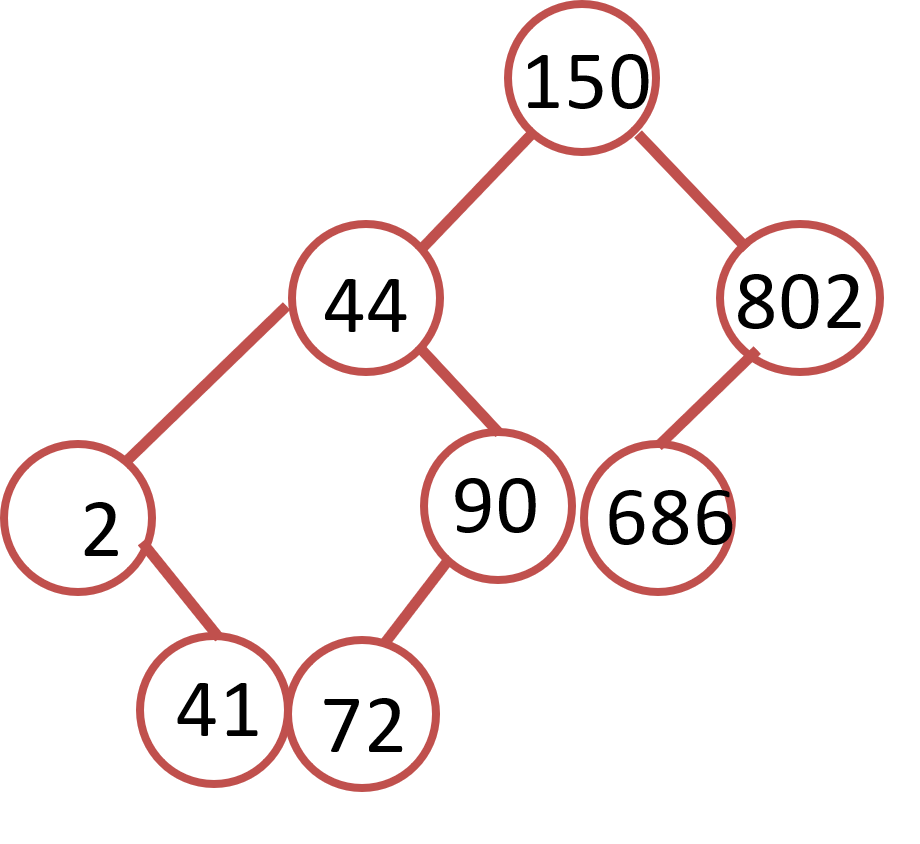
686 (802)

After 3nd pass, left subarray:

(2) 41

Right subarray:

72 (90)



Final sorted: 2 41 44 72 90 150 686 802

14. (5 pts) Lecture 14-radix sort

Sort this array of numbers with Radix sort, with radix of 10, into ascending order. Show the intermediate results after each pass, and give the final sorted array.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 150 | 41 | 72 | 90 | 802 | 44 | 2 | 686 |

After 1st pass

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

After 2nd pass

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

After 3rd pass

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

ANS:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 150 | 41 | 72 | 90 | 802 | 44 | 2 | 686 |

After 1st pass (sorting by the last digit)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 150 | 90 | 41 | 72 | 802 | 2 | 44 | 686 |

After 2nd pass (sorting by the 2nd to last digit)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 802 | 2 | 41 | 44 | 150 | 72 | 686 | 90 |

After 3rd pass (sorting by the first digit)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 41 | 44 | 72 | 90 | 150 | 686 | 802 |