**CSC017 Fall 2024 Final Exam Sample Questions**

**Student Name： ID：**

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

Each multiple choice question has exactly one correct choice as answer. If there are multiple choices that are correct, please select the choice “All of the above”. If there are no choice that is correct, please select the choice “None of the above”.

Lec 4. Which of the following regular expressions if any, is equivalent to (a|b)\*a+

A. (a|b)\*b(a)+

B. a\*b\*a+

C. (a|b)\*aa\*

D. (a\*b\*)\*aa+

ANS: C.

All strings of a’s and b’s that end with a run of one or more a’s. (This explanation is not necessary for the exam.)

Lec 4. Which of the following regular expressions if any, is equivalent to (a|b|c)\*

A. [a-c]\*

B. [a]\*[bc]\*

C. [abc]+

D. (a|b|c)(a|b|c)\*

ANS: A.

All strings consisting of 0 or more a’s, b’s, and c’s in any order. (This explanation is not necessary for the exam.)

Lec 7. Insert the following keys into the hash tables below, where the hash function is modulo table size (%TableSize). What is the load factor? [1 point] In the top table, resolve collisions with linear probing. In the middle table, resolve collisions with quadratic probing. In the bottom table resolve by separate chaining into a sorted linked list (with the smallest element at the head of the list). If an insertion fails, record which key failed, but attempt to insert any later keys in the list. Do not resize the tables. Insert these keys: 22, 14, 32, 42, 37, 13

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

1) with linear probing:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  | 22 | 32 | 14 | 42 | 13 | 37 |  |  |

Steps for Insertion (You do not need to explain the steps in the exam):

1. Key = 22: Hash index = 2. Index 2 is empty, so insert 22 at index 2.

2. Key = 14: Hash index = 4. Index 4 is empty, so insert 14 at index 4.

3. Key = 32: Hash index = 2. Index 2 is occupied (collision with key 22), so apply linear probing. Check index (2 + 1) % 10 = 3, which is empty. Insert 32 at index 3.

4. Key = 42: Hash index = 2. Index 2 is occupied (collision with key 22), so apply linear probing. Check index (2 + 1) % 10 = 3, which is occupied. Check index (2 + 2) % 10 = 4, which is occupied. Check index (2 + 3) % 10 = 5, which is empty. Insert 42 at index 5.

5. Key = 37: Hash index = 7. Index 7 is empty, so insert 37 at index 7.

6. Key = 13: Hash index = 3. Index 3 is occupied (collision with key 32), so apply linear probing. Check index (3 + 1) % 10 = 4, which is occupied. Check index (3 + 2) % 10 = 5, which is occupied. Check index (3 + 3) % 10 = 6, which is empty. Insert 13 at index 6.

2) with quadratic probing: (If the primary hash index is x, probes go to x+1, x+4, x+9, x+16, x+25 and so on, this results in Secondary Clustering.)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  | 22 | 32 | 14 |  | 42 | 37 |  | 13 |

Steps for Insertion (You do not need to explain the steps in the exam):

1. Key = 22: Hash index = 2. Index 2 is empty, so insert 22 at index 2.

2. Key = 14: Hash index = 4. Index 4 is empty, so insert 14 at index 4.

3. Key = 32: Hash index = 2. Index 2 is occupied (collision with key 22), so apply quadratic probing. Check index (2 + 1^2) % 10 = 3, which is empty. Insert 32 at index 3.

4. Key = 42: Hash index = 2. Index 2 is occupied (collision with key 22), so apply linear probing. Check index (2 + 1^2) % 10 = 3, which is occupied. Check index (2 + 2^2) % 10 = 6, which is empty. Insert 42 at index 6.

5. Key = 37: Hash index = 7. Index 7 is empty, so insert 37 at index 7.

6. Key = 13: Hash index = 3. Index 3 is occupied (collision with key 32), so apply linear probing. Check index (3 + 1^2) % 10 = 4, which is occupied. Check index (3 + 2^2) % 10 = 7, which is occupied. Check index (3 + 3^2) % 10 = 2, which is occupied. Check index (3 + 4^2) % 10 = 9, which is empty. Insert 13 at index 9.

3) with separate chaining:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  | 22,32,42 | 13 | 14 |  |  | 37 |  |  |

Lec 8. Which of the following statements are true about the following binary search tree? Select all that apply.

21

9

6

20

56

45

15

A. 9 may be inserted before or after 56

B. There may be multiple insertion orders that could have produced this tree.

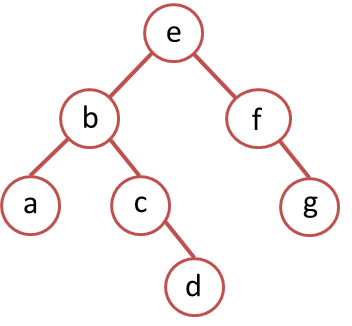
C. 21 must have been inserted first.

D. 9 must have been inserted before 15

E. All of the above

ANS: E

Lec 8. Give Preorder, In-order, Post-order, and Level-order traversal (by BFS) of this tree:



ANS: Preorder: ebacdfg

In-order: abcdefg

Post-order: adcbgfe

Level-order: ebfacgd

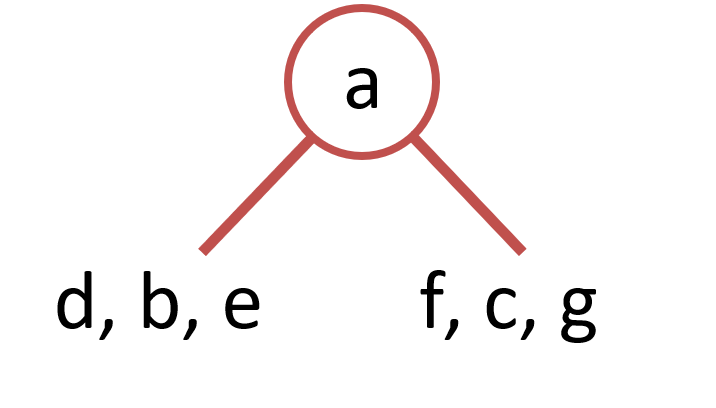
Lec. 8. The in-order and pre-order traversal of a binary tree are: dbeafcg and abdecfg, respectively. The post-order traversal of this binary tree is:

ANS: debfgca

The pre-order traversal gives us the root node first. Using this root, we divide the in-order traversal into left and right subtrees:

- The first element of the pre-order traversal is `a`, which is the root.

- In the in-order traversal, everything to the left of `a` (`dbe`) forms the left subtree, and everything to the right (`fcg`) forms the right subtree.



Left Subtree

- Pre-order for left subtree: `bde` (from pre-order after root `a`).

- In-order for left subtree: `dbe` (from in-order before root `a`).

- Root of this subtree is `b` (first element of pre-order for this subtree).

- Split `dbe` into:

- Left subtree: `d` (in-order before root `b`).

- Right subtree: `e` (in-order after root `b`).

Right Subtree

- Pre-order for right subtree: `cfg` (from pre-order after processing left subtree).

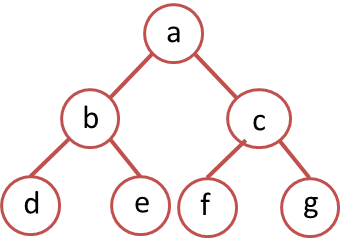
- In-order for right subtree: `fcg` (from in-order after root `a`).

- Root of this subtree is `c`.

- Split `fcg` into:

- Left subtree: `f`.

- Right subtree: `g`.



Post-order traversal follows the order (Left, Right, Root). Using the reconstructed tree:

1. Traverse the left subtree (`d`, `e`, then `b` as root).

2. Traverse the right subtree (`f`, `g`, then `c` as root).

3. Visit the main root (`a`).

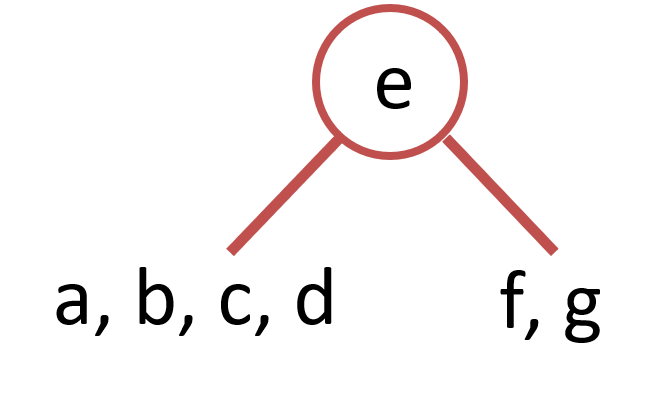
Thus, combining these steps gives the post-order traversal: debfgca.

Lec 8. The in-order and post-order traversal of a binary tree are: abcdefg and adcbgfe, respectively. The pre-order traversal of this binary tree is:

ANS: The post-order traversal gives us the root node last. Using this root, we divide the in-order traversal into left and right subtrees:

- The last element of the post-order traversal is `e`, which is the root.

- In the in-order traversal, everything to the left of `e` (`abcd`) forms the left subtree, and everything to the right (`fg`) forms the right subtree.



Left Subtree

- Post-order for left subtree: `adcb` (from post-order before root `e`).

- In-order for left subtree: `abcd` (from in-order before root `e`).

- Root of this subtree is `b` (last element of post-order for this subtree). Left subtree contains a, right subtree contains cd.

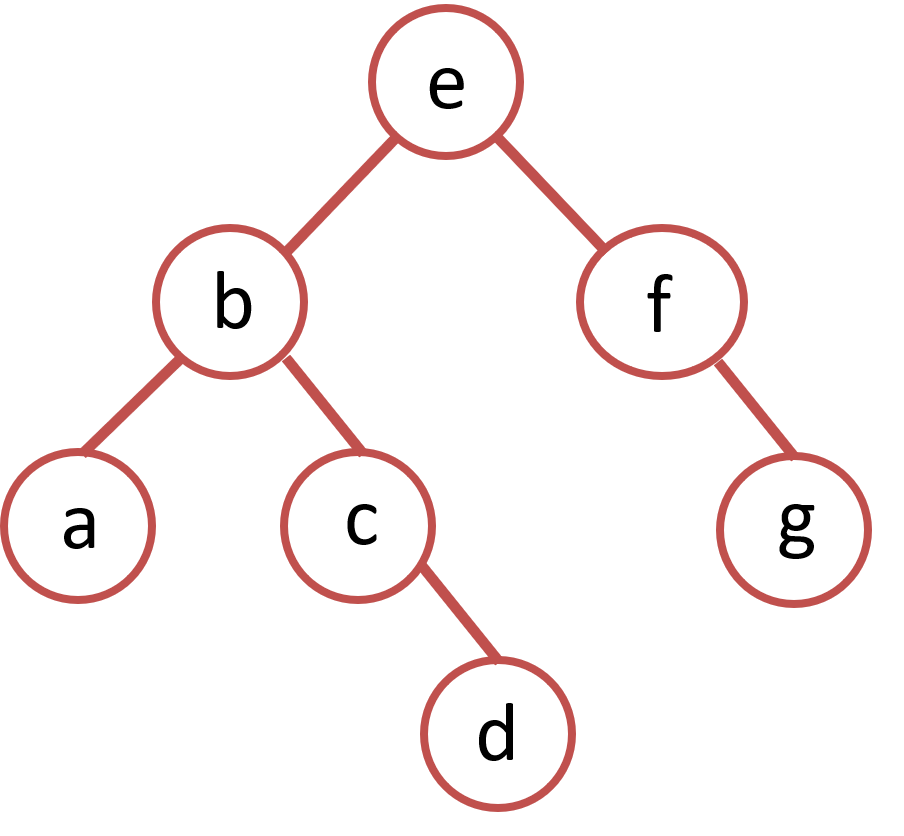
Post-order for the right subtree of cd is `dc`, hence c is the root.

In-order for the right subtree of cd is `cd`, hence d is c’s right child.

Right Subtree

Post-order for the right subtree of cd is `gf`, hence f is the root.

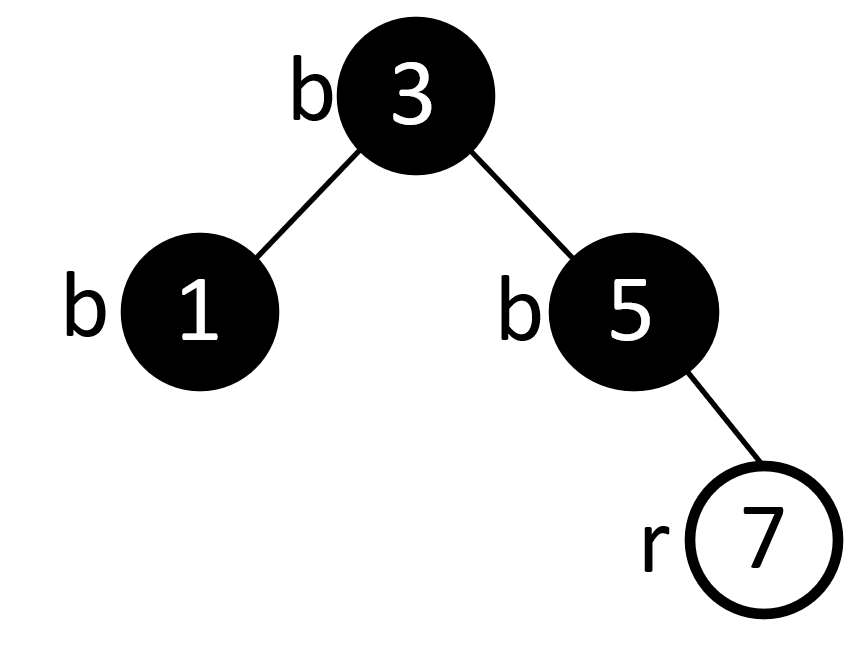
In-order for the right subtree of cd is `fg`, hence g is f’s right child.

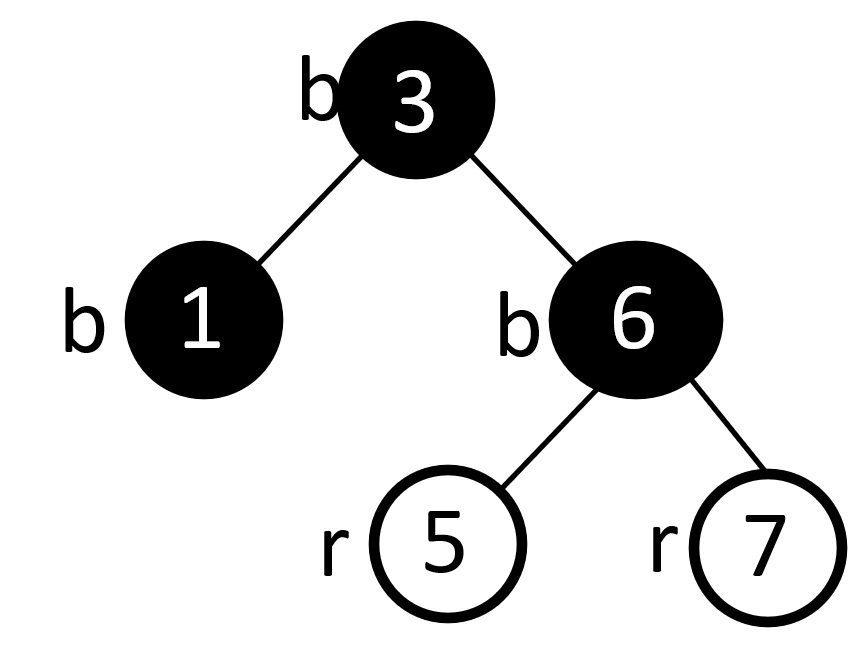


Pre-order traversal follows the order (Root, Left, Right)

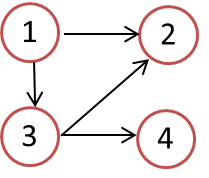
Pre-order traversal: ebacdfg.

Lec 9. Consider the following red-black tree. Draw the resulting red-black tree after inserting 6. (Due to lack of color printer, I use black filled circles to denote black nodes, and white circiles to denote red nodes. You may put an annotation beside each node, using r for red, b for black)



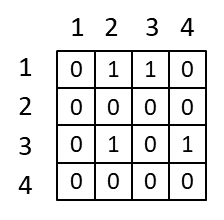
ANS: 

Lec. 10. 1) Write out the adjacency list and adjacency matrix representations of the following digraph. 2) Write out the adjacency list and adjacency matrix representations of 2-hop neighbors of the following digraph.



ANS:

Adjacency matrix:



Adjacency list:

1 → {2, 3}

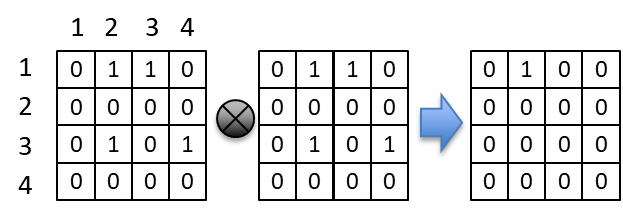
2 → {}

3 → {2, 4}

4 → {}

2-hop neighbors adjacency matrix:

Matrix-matrix multiplication gives: (There is only one 2-hop path from 1 to 2.)



2-hop neighbors adjacency list:

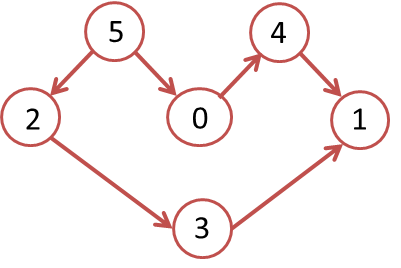
1 → {2}

2 → {}

3 → {}

4 → {}

Lec. 10. Find a topological sort of the following graph by running depth-first search DFS ´*starting from node 5*, then return vertices in reverse postorder traversal.

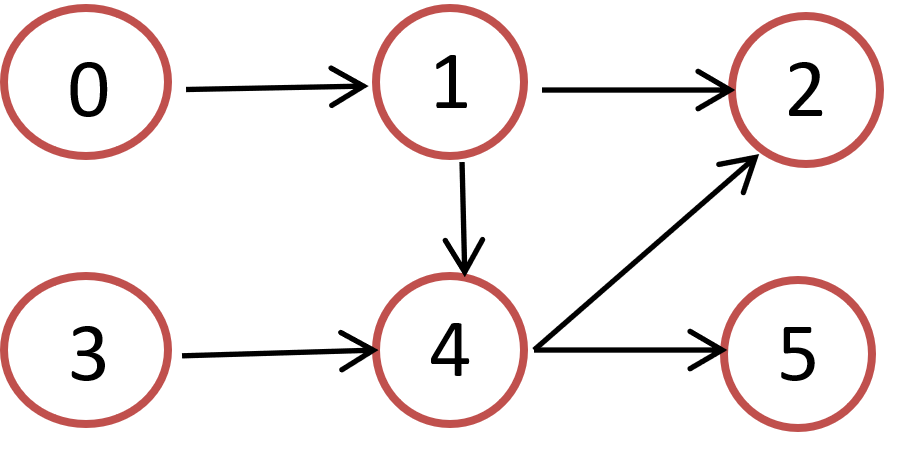


ANS: DFS pre-order traversal: 5 2 3 1 0 4

DFS post-order traversal: 1 3 2 4 0 5

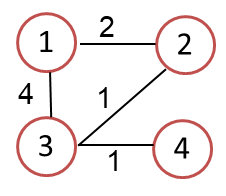
Reverse post-order traversal: 5 0 4 2 3 1

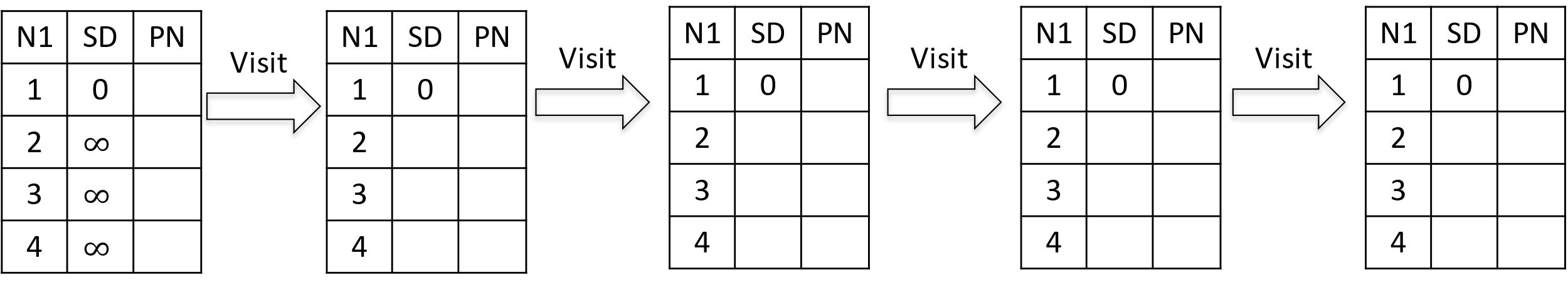
Lec. 10. Find a topological sort of the following graph by running Kahn’s algorithm starting from node 0. (There may be multiple possible topological sorts, and you just need to give one of them.)



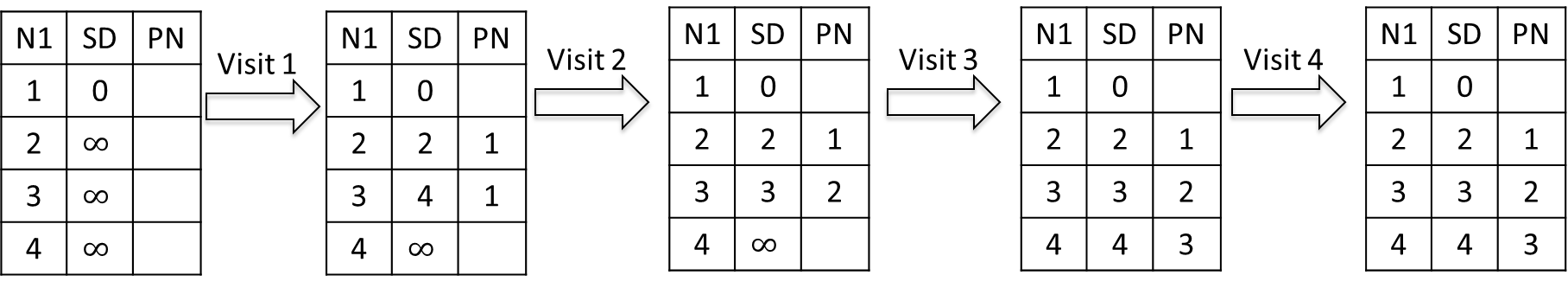
ANS: 0 3 1 4 2 5 or 0 3 1 4 5 2 or 0 1 3 4 2 5 or 0 1 3 4 5 2

Lec. 11. Use Dijkstra’s algorithm to find shortest paths starting from source vertex 1 for the following undirected graph. SD: Shortest Distance. PN: Previous Node

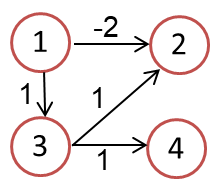




ANS:



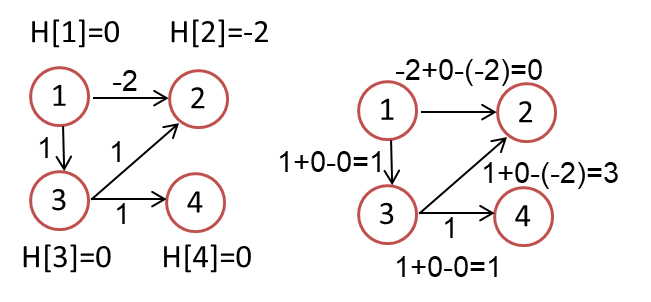
Lec. 11. Consider the following weighted digraph. As part of Johnson’s algorithm for All-pairs Shortest Paths, add a dummy source vertex d, and add edges with weight 0 from d to all vertices of G. Let the modified graph be G’.  Compute the shortest distances (instead of running Bellman-Ford algorithm, you can compute it 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.



ANS: H[1]=0, H[2]=-2, H[3]=0, H[4]=0

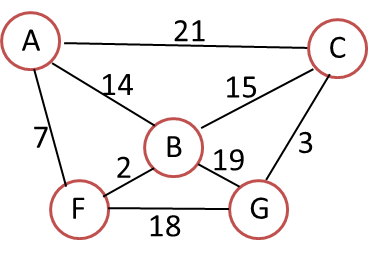
W[1][2]=-2+0-(-2)=0, W[1][3]=1+0-0=1, W[3][2]=1+0-(-2)=3, W[1][4]=1+0-0=1

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



Lec 12. Minimum spanning tree.

1) 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:

AF, BF, BC, CG

2) List the edges in the order which Kruskal’s algorithm includes them into the MST:

ANS:

BF, CG, AF, BC

Lec13. Is this a complete binary tree?

A group of white circles on a black background

Description automatically generated

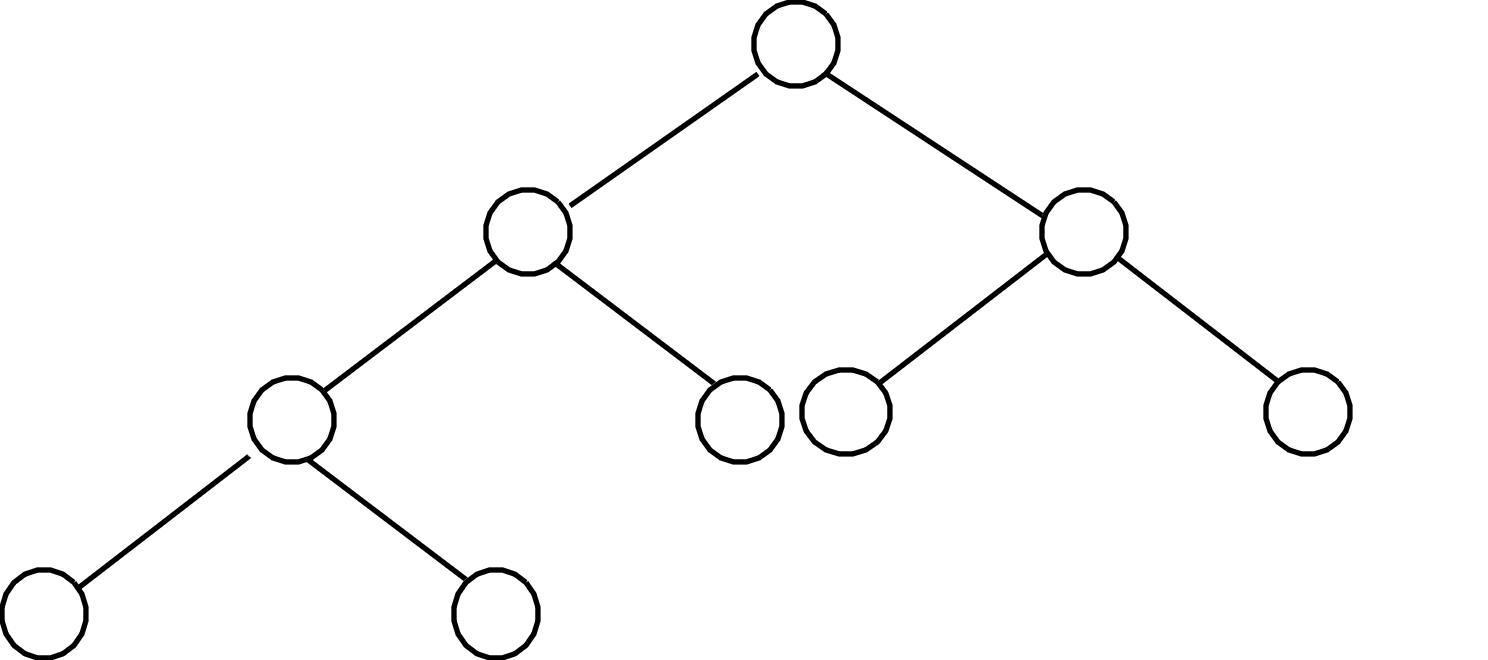
A. Yes

B. No

ANS: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

B

Lec13. Is this a complete binary tree?



A. Yes

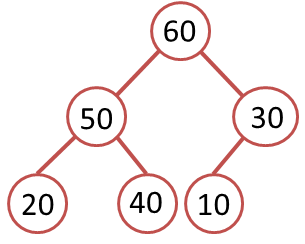
B. No

ANS: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A

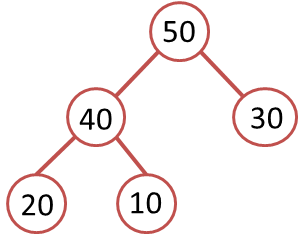
Lec13. Draw the max heap tree constructed by adding each element in this order: 40, 60, 10, 20, 50, 30.

ANS:



Refer to this video: Max Heap Animations | Data Structure | Visual How <https://www.youtube.com/watch?v=uU0iWaVxMgc>

Lec13. Draw the max heap tree after removing the root 60 from the max heap tree.



Lec 13. The first row below shows a list of numbers to be sorted using mergesort. Show the split into sublists, then show the merge steps. When there are an odd number of elements in a list, make the left sublist larger. Put an ‘X’ on any sublist you don’t use.

65 17 19 85 97 12 23

\_ \_ \_ \_ \_ \_ \_ \_

ANS: A white background with numbers

Description automatically generated

Lec. 13. The first row below shows a list of numbers to be sorted using quicksort. Use *the first number* of each sublist as the pivot. In the second row of the chart, enter the pivot in the circle. Then enter the numbers in the left and right sublists to the left and right of the pivot, respectively. (Note that the two sublists need not be the same size.) Repeat this process on every line. If a sublist is empty, just draw an X on it. Draw the coresponding Binary Search Tree. Give the final sorted list.

**65 17 19 85 97 12 23**

ANS:

**65 17 19 85 97 12 23**

65

65

**17 19 12 23 85 97**

12 **19 23 X 97**

17

85

85

17

**X**

12

**X**

**X** 19

**23**

**X**

X

**X**

**X** 97

**X**

65

17

12

19

23

85

97

Sorted list: 12 17 19 23 65 85 97

Lec 14. Sort this array of numbers with Radix sort, with radix of 10, into acending order. Show the intermediate results after each pass.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 170 | 45 | 75 | 90 | 802 | 24 | 2 | 66 |

After 1st pass

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

After 2nd pass

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

After 3rd pass

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

ANS:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 170 | 45 | 75 | 90 | 802 | 24 | 2 | 66 |

After 1st pass (sorting by the last digit)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 170 | 90 | 802 | 2 | 24 | 45 | 75 | 66 |

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 802 | 2 | 24 | 45 | 66 | 170 | 75 | 90 |

After 3rd pass (sorting by the first digit)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 24 | 45 | 66 | 75 | 90 | 170 | 802 |