Sunbeam Institute of Information Technology

Pune and Karad

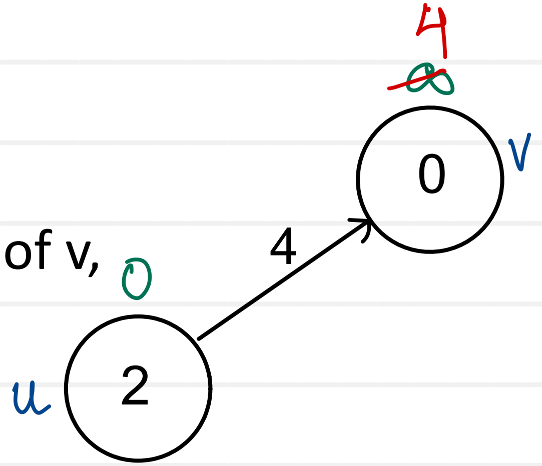
Module – Data Structures and Algorithms

Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

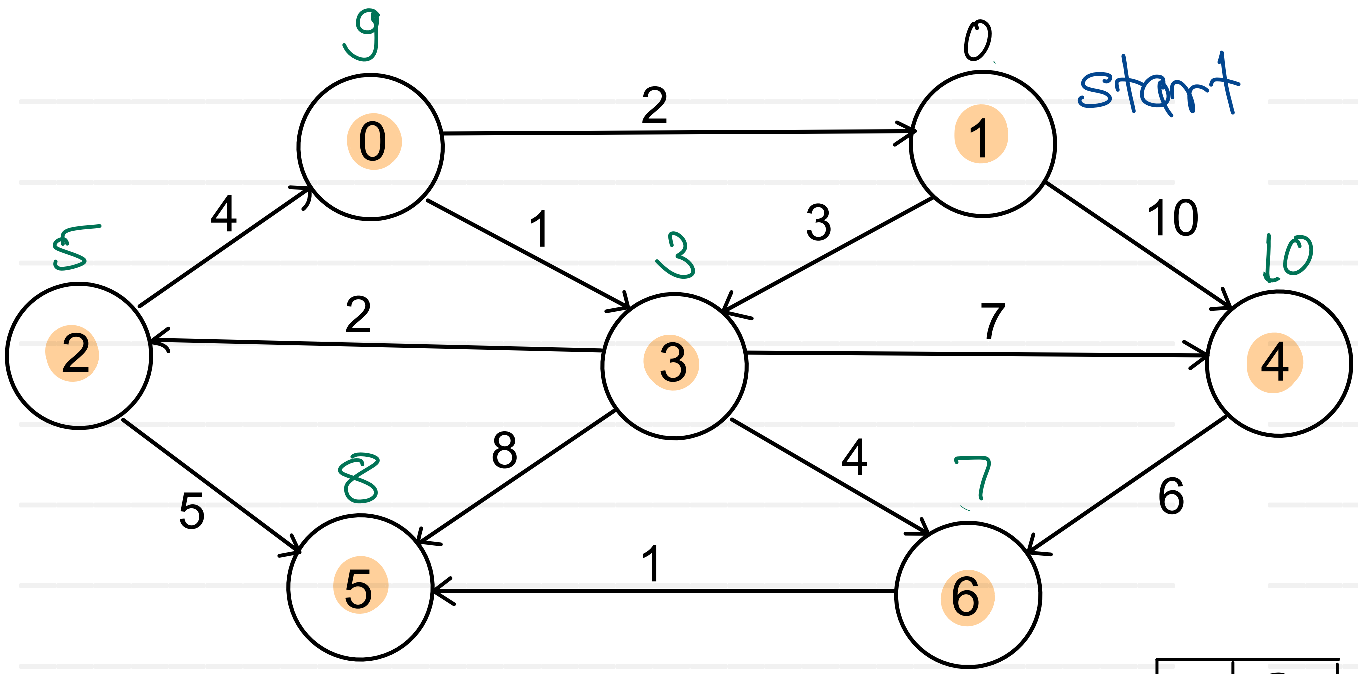
Dijkstra’s Algorithm

1. Create a set spt to keep track of vertices included in shortest path tree.
2. Track distance of all vertices in the input graph. Distance for all vertices should be initialized to INF. The start vertex distance should be 0.
3. While spt doesn't include all the vertices
   i. Pick a vertex u which is not there in spt and has minimum distance.
   ii. Include vertex u to spt.
   iii. Update distances of all adjacent vertices of u.
   For each adjacent vertex v,
      if distance of u + weight of edge u-v is less than the current distance of v,
      then update its distance as distance of u + weight of edge u-v.




```
if( dist[u]+ adjmat[u][v]  < dist[v])
    dist[v] = dist[u] +adjmat [u][v]
```

Dijkstra's Algorithm



	D
0	9
1	0
2	5
3	3
4	10
5	8
6	7

	D
0	∞
1	0
2	∞
3	3
4	10
5	∞
6	∞

	D
0	∞
1	0
2	5
3	3
4	10
5	11
6	7

	D
0	9
1	0
2	5
3	3
4	10
5	10
6	7

	D
0	9
1	0
2	5
3	3
4	10
5	8
6	7

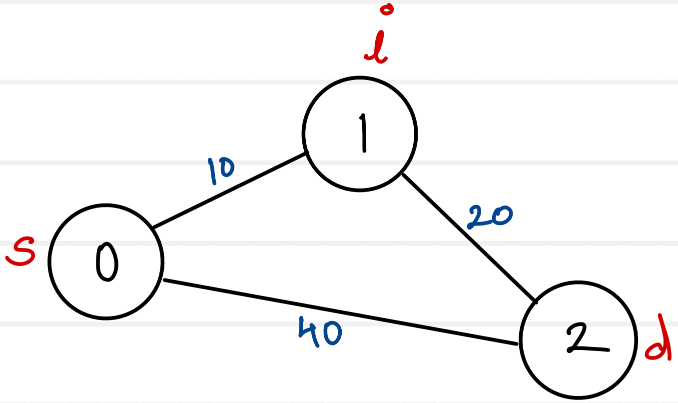
	D
0	9
1	0
2	5
3	3
4	10
5	8
6	7

	D
0	9
1	0
2	5
3	3
4	10
5	8
6	7

Floyd Warshall Algorithm

1. Create distance matrix to keep distance of every vertex from each vertex.
   Initially assign it with weights of all edges among vertices
   (i.e. adjacency matrix).

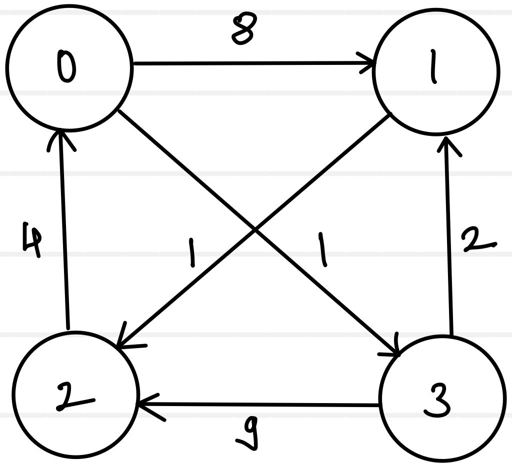
2. Consider each vertex (i) in between pair of any two vertices (u, v) and
   find the optimal distance between s & d considering intermediate vertex
   i.e. dist(u,v) = dist(u,i) + dist(i,v),
   if dist(u,i) + dist(i,v) < dist(u,v).



dist via i　　　　direct dist

```
if(dist[s][i] + dist[i][d] < dist[s][d])
    dist[s][d] = dist[s][i] + dist[i][d];
```

Floyd Warshall Algorithm



$$\begin{array}{cc} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \left[\begin{array}{cccc} \infty & 8 & \infty & 1 \\ \infty & \infty & 1 & \infty \\ 4 & \infty & \infty & \infty \\ \infty & 2 & 9 & \infty \end{array}\right] \end{array}$$

$$d = \begin{array}{cc} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \left[\begin{array}{cccc} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{array}\right] \end{array}$$

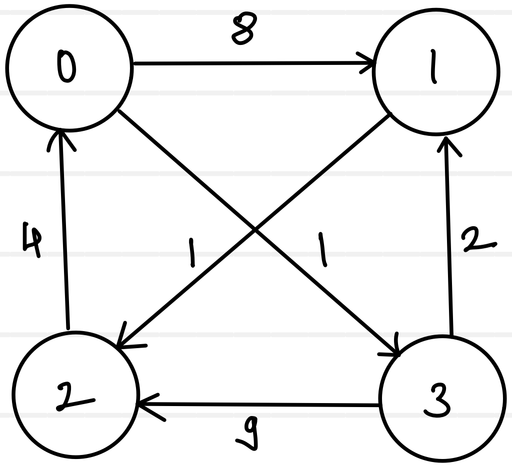
$$d_0 = \begin{array}{cc} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \left[\begin{array}{cccc} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{array}\right] \end{array}$$

$$d_1 = \begin{array}{cc} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \left[\begin{array}{cccc} 0 & 8 & 9 & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 3 & 0 \end{array}\right] \end{array}$$

$$d_2 = \begin{array}{cc} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \left[\begin{array}{cccc} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{array}\right] \end{array}$$

$$d_3 = \begin{array}{cc} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \left[\begin{array}{cccc} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{array}\right] \end{array}$$

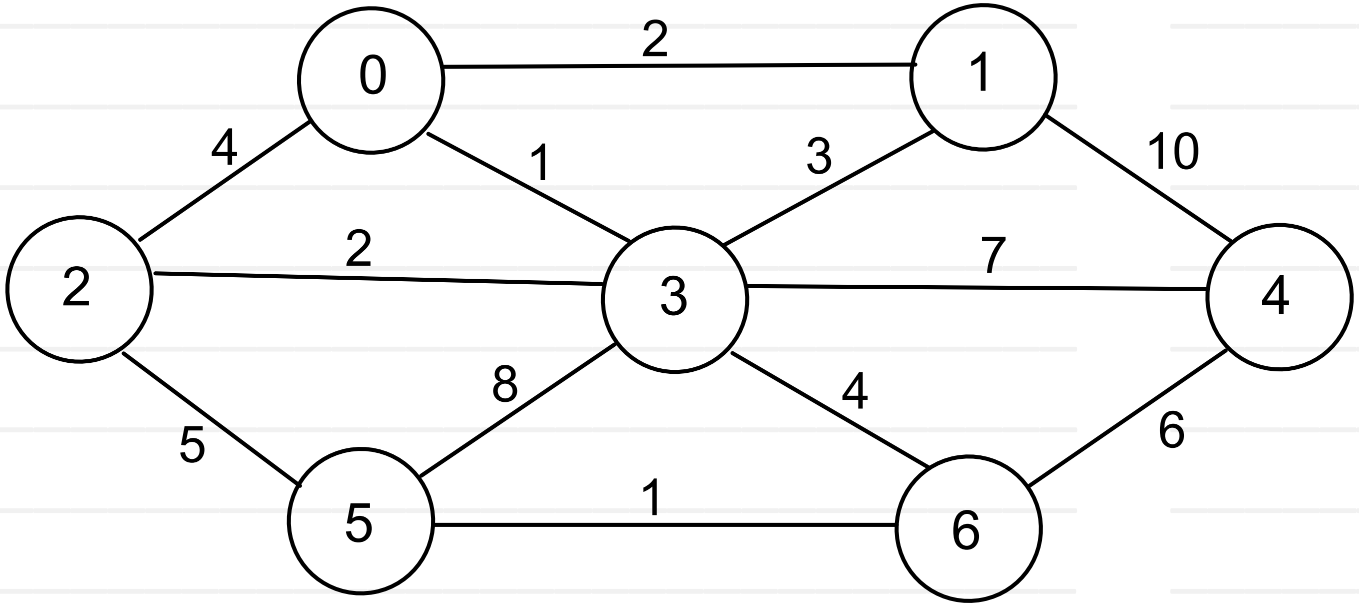
Floyd Warshall Algorithm




```
void floydWarshallAlgorithm( ) {
    int dist[][] = new int[v][v];
    for(int s=0 ; s<v ; s++) {
        for(int d=0 ; d<v ; d++)
            dist[s][d] = adjmat[s][d];
        dist[s][s] = 0 ;
    }
    for(int i=0 ; i<v ; i++) {
        for (int s=0; s<v ; s++) {
            for ( int d=0; d<v ; d++) {
                if(dist[s][i] + dist[i][d] < dist[s][d])
                    dist[s][d] = dist[s][i] + dist[i][d];
            }
        }
    }
}
```

Union Find Algorithm

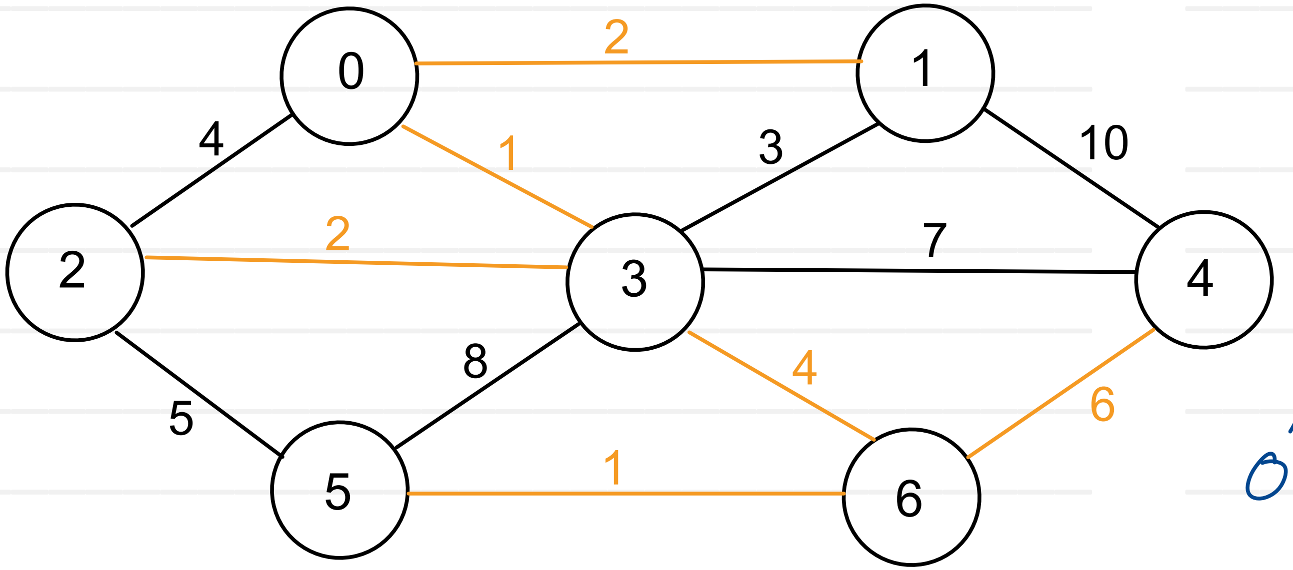
1. Consider all vertices as disjoint sets (parent = -1).
2. For each edge in the graph
    1. Find set(root) of first vertex.
    2. Find set(root) of second vertex.
    3. If both are in same set(same root), cycle is detected.
    4. Otherwise, merge(Union) both the sets i.e. add root of first set under second set

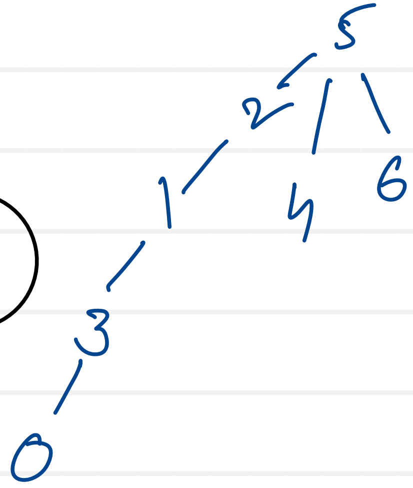


Union Find Algorithm

Parents

0	1	2	3	4	5	6
3	2	5	1	5	-1	5





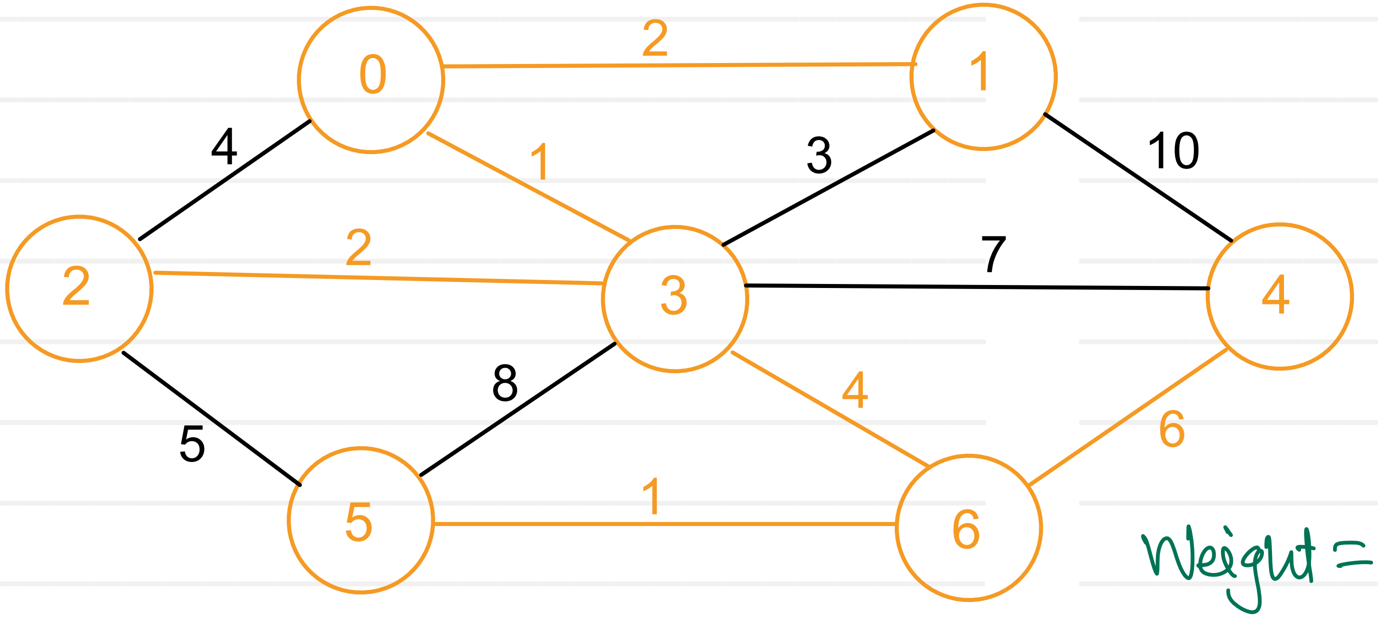
sr	dr	s	d	
0	3	0	3	- 1
6	5	6	5	- 1
3	1	0	1	- 2
1	2	3	2	- 2
2 ==	2	1	3	- 3
2 ==	2	2	0	- 4
2	5	3	6	- 4
5 ==	5	2	5	- 5
4	5	4	6	- 6
		3	4	- 7
		3	5	- 8
		1	4	- 1

```
int find(int v, int parents[]){
    while(parents[v] != -1)
        v = parents[v];
    return v;
}
```

```
void union(int sr, int dr, int parents[]){
    parents[sr] = dr;
              0    3
}
```

Kruskal's Algorithm

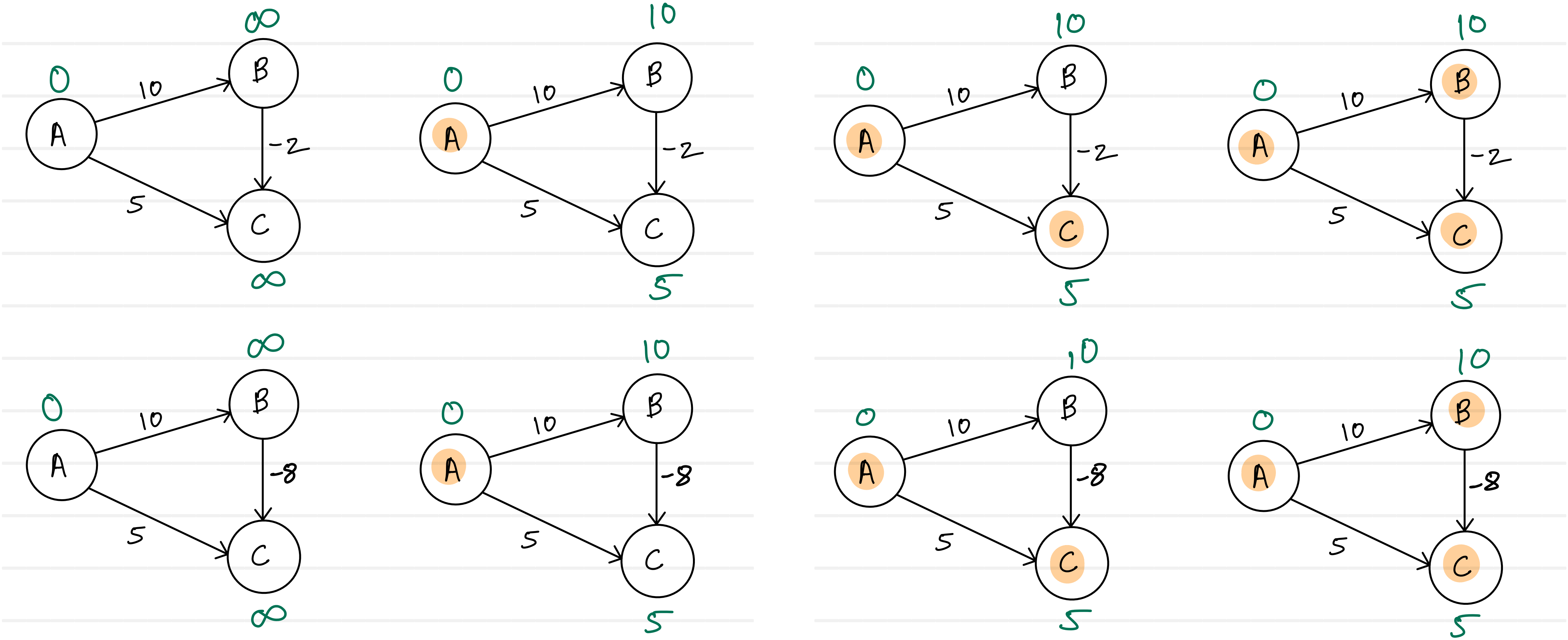
1. Sort all the edges in ascending order of their weight.
2. Pick the smallest edge.
   Check if it forms a cycle with the spanning tree formed so far.
   If cycle is not formed, include this edge.
   Else, discard it.
3. Repeat step 2 until there are (V-1) edges in the spanning tree.



0 3 - 1 ✓
6 5 - 1 ✓
0 1 - 2 ✓
3 2 - 2 ✓
1 3 - 3 ✗
2 0 - 4 ✗
3 6 - 4 ✓
2 5 - 5 ✗
4 6 - 6 ✓
3 4 - 7
3 5 - 8
1 4 - 10

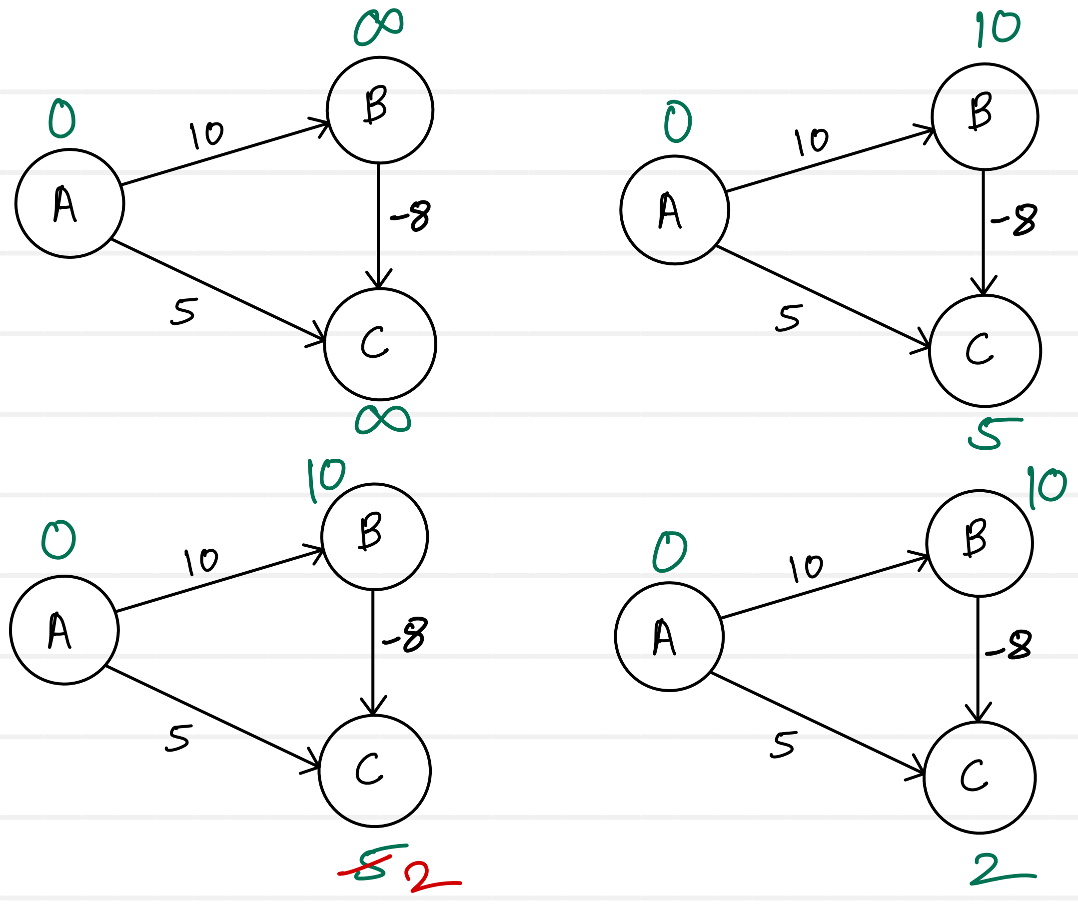
Weight = 16

Dijkstra’s Algorithm



Bellman Ford Algorithm

1. Initializes distances from the source to all vertices as infinite and distance to the source itself as 0.
2. Calculates shortest distance V-1 times:
   For each edge u-v,
   if dist[v] > dist[u] + weight of edge u-v,
   then update dist[v], so that
      dist[v] = dist[u] + weight of edge u-v.
3. Check if negative edge cycle in the graph:
   For each edge u-v,
   if dist[v] > dist[u] + weight of edge (u,v),
   then graph has -ve weight cycle.



Graph applications

- Graph represents flow of computation/tasks. It is used for resource planning and scheduling. MST algorithms are used for resource conservation. DAG are used for scheduling in Spark or Tez.

- In OS, process and resources are treated as vertices and their usage is treated as edges. This resource allocation algorithm is used to detect deadlock.

- In social networking sites, each person is a vertex and their connection is an edge. In Facebook person search or friend suggestion algorithms use graph concepts.

- In world wide web, web pages are like vertices; while links represents edges. This concept can be used at multiple places.
    - Making sitemap
    - Downloading website or resources
    - Developing web crawlers
    - Google page-rank algorithm

- Maps uses graphs for showing routes and finding shortest paths. Intersection of two (or more) roads is considered as vertex and the road connecting two vertices is considered to be an edge.

Merge sort

1. Divide array in two parts
2. Sort both partitions individually (by merge sort only)
3. Merge sorted partitions into temporary array
4. Overwrite temporary array into original array

6	1	9	7	3	8	2	4	5
0	1	2	3	4	5	6	7	8
l				m				r

6	1	9	7	3
0	1	2	3	4

8	2	4	5
5	6	7	8

i

1	3	6	7	9
0	1	2	3	4

j

2	4	5	8
5	6	7	8

k

temp

1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8

No. of elements = n

No. of levels = $\log n$

Comps per level $\propto n$

Total comps = $n \log n$

Time $\propto$ comps

Time $\propto n \log n$

Best, Avg, Worst: $\boxed{T(n) = O(n \log n)}$

To merge sorted partitions we need temp array

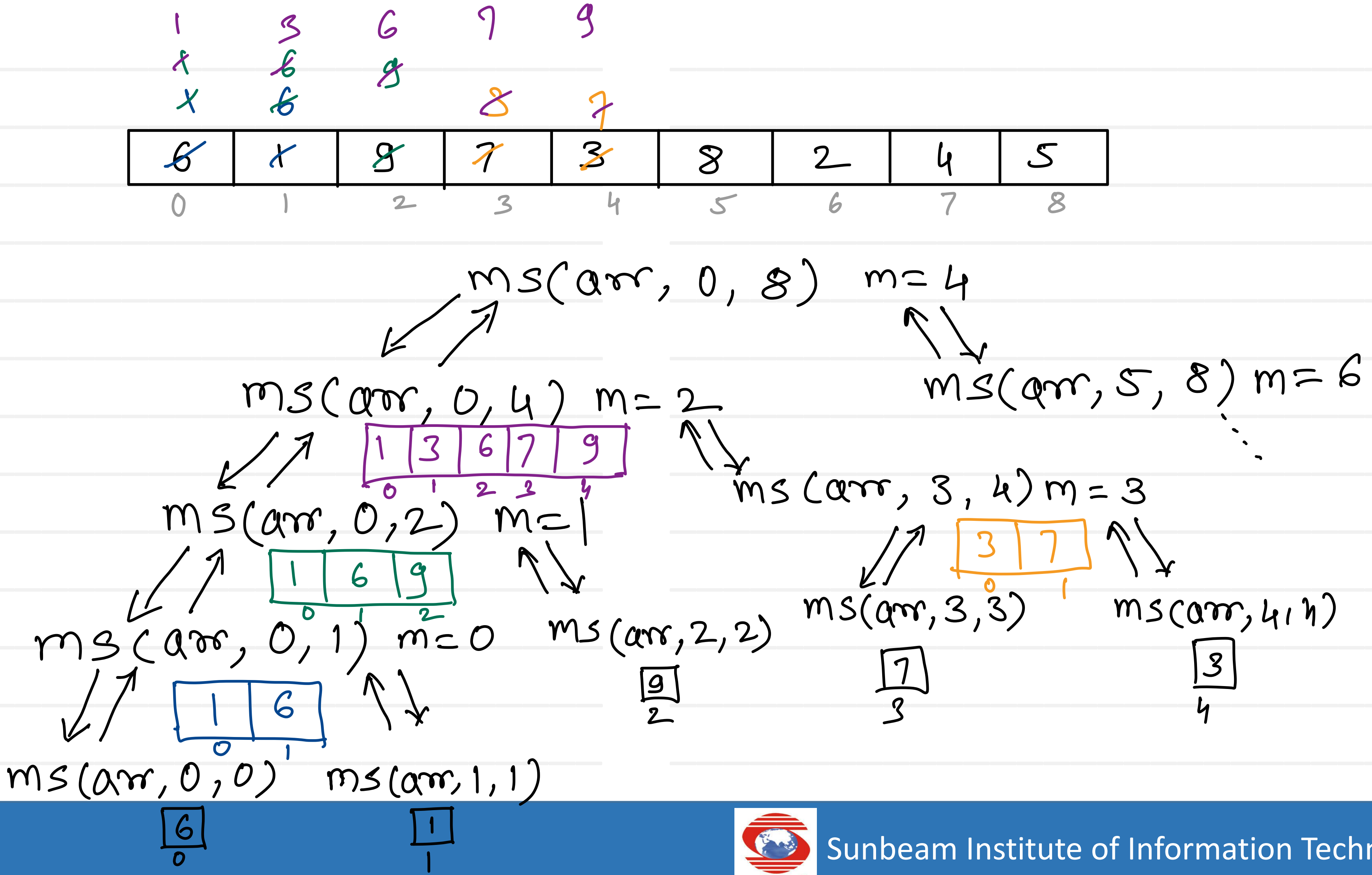
temp → Processing variable

$AS \propto$ sizeof(temp)

$AS \propto n$

$\boxed{AS(n) = O(n)}$

Merge sort



1 3 6 7 9
8 7
6 1 9 7 3 8 2 4 5
0 1 2 3 4 5 6 7 8
ms(arr, 0, 8) m=4
ms(arr, 0, 4) m=2
1 3 6 7 9
ms(arr, 5, 8) m=6
ms(arr, 0, 2) m=1
1 6 9
ms(arr, 3, 4) m=3
3 7
ms(arr, 0, 1) m=0
1 6
ms(arr, 2, 2)
ms(arr, 3, 3)
ms(arr, 4, 4)
ms(arr, 0, 0)
ms(arr, 1, 1)

Quick sort

1. Select pivot/axis/reference element from array → selection of pivot :
2. Arrange lesser elements on left side of pivot
3. Arrange greater elements on right side of pivot
4. Sort left and right side of pivot again (by quick sort)

selection of pivot :
1) extreme left or right
2) middle element
3) median ← random 3, random 5

no. of elements = n
no. of levels = log n
comps per level = n
Total comps = n log n
Time ∝ n log n

Best, Avg: $\boxed{T(n) = O(n \log n)}$

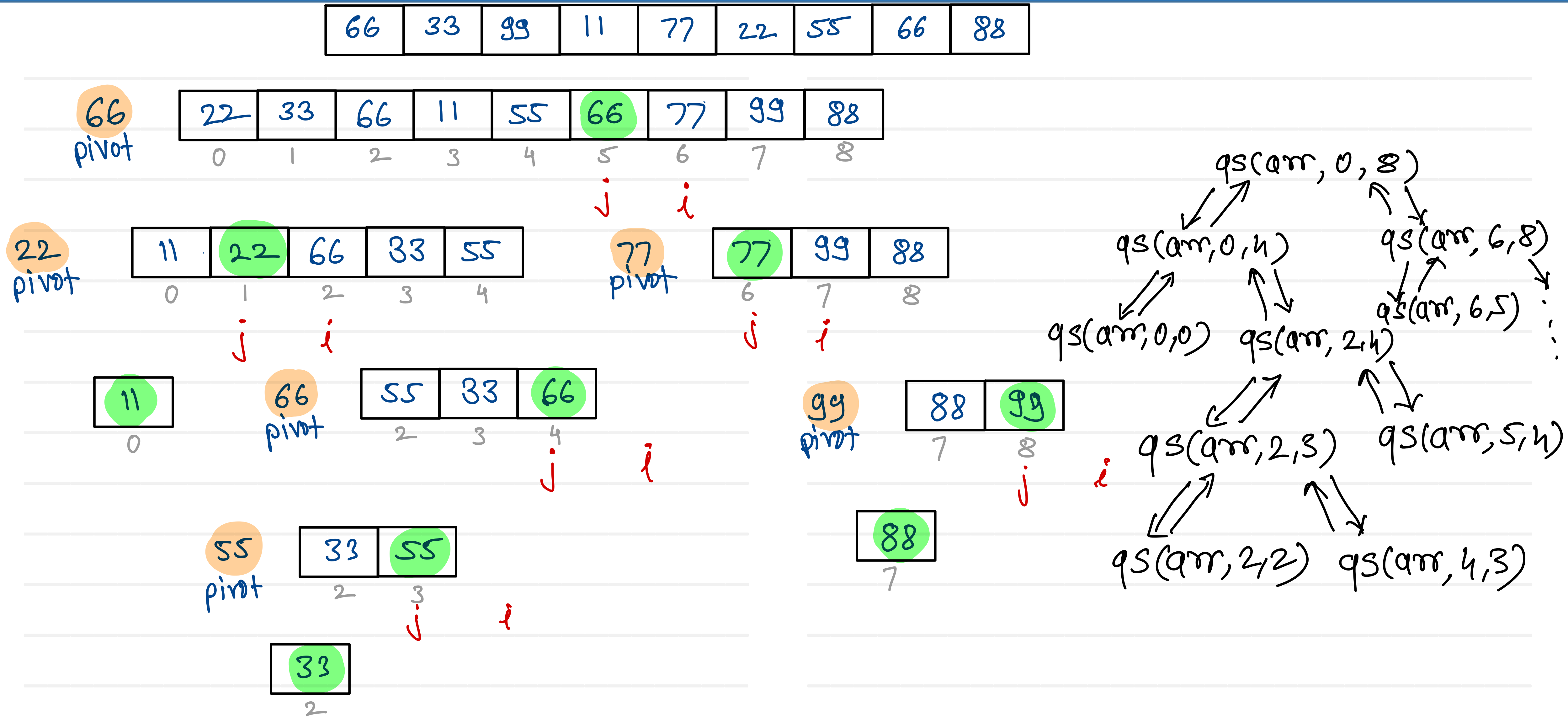
11	22	33	44	55
	22	33	44	55
		33	44	55
			44	55
				55

no. of level ∝ n
per level comps = n
comps = n^2

$\boxed{T(n) = O(n^2)}$ worst

- Time complexity of quick sort is dependent on selection of pivot

Quick sort



66 33 99 11 77 22 55 66 88
66 pivot
22 33 66 11 55 66 77 99 88
0 1 2 3 4 5 6 7 8
j i
22 pivot
11 22 66 33 55
0 1 2 3 4
j i
77 pivot
77 99 88
6 7 8
j i
11
0
66 pivot
55 33 66
2 3 4
j i
99 pivot
88 99
7 8
j i
88
7
55 pivot
33 55
2 3
j i
33
2
qs(arr, 0, 8)
qs(arr, 0, 4)
qs(arr, 6, 8)
qs(arr, 0, 0)
qs(arr, 2, 4)
qs(arr, 6, 5)
qs(arr, 2, 3)
qs(arr, 5, 4)
qs(arr, 2, 2)
qs(arr, 4, 3)

Hashing

Array : linear search – $O(n)$
binary search – $O(\log n)$

Linked List : search – $O(n)$

Binary tree : search – $O(n)$
BST : search – $O(\log n)$

Graph : search – $O(V^2)$

Hash Table : search – $O(1)$

- hashing is a technique in which data can be inserted, deleted and searched in constant average time O(1)
- Implementation of hashing is known as hash table
- Hash table is array of fixed size in which elements are stored in key - value pairs

Array - Hash table
Index - Slot

- In hash table only unique keys are stored
- Every key is mapped with one slot of the table and this is done with the help of mathematical function known as hash function

Hashing

Key value

8-V1
3-V2
10-V3
4-V4
6-V5
13-V6

collision

size=10

10, V3	0
	1
	2
3, V2	3
4, V4	4
	5
6, V5	6
	7
8, V1	8
	9

Hash Table

h(k) = k % size

h(8) = 8 % 10 = 8
h(3) = 3 % 10 = 3
h(10) = 10 % 10 = 0
h(4) = 4 % 10 = 4
h(6) = 6 % 10 = 6
h(13) = 13 % 10 = 3

Collision :
whenever multiple keys yield same slot

Collision handling techniques:
1. Closed addressing
2. Open addressing
   i. Linear probing
   ii. Quadratic probing
   iii. Double hashing

Add : O(1)
1. slot = h(k)
2. arr[slot] = data

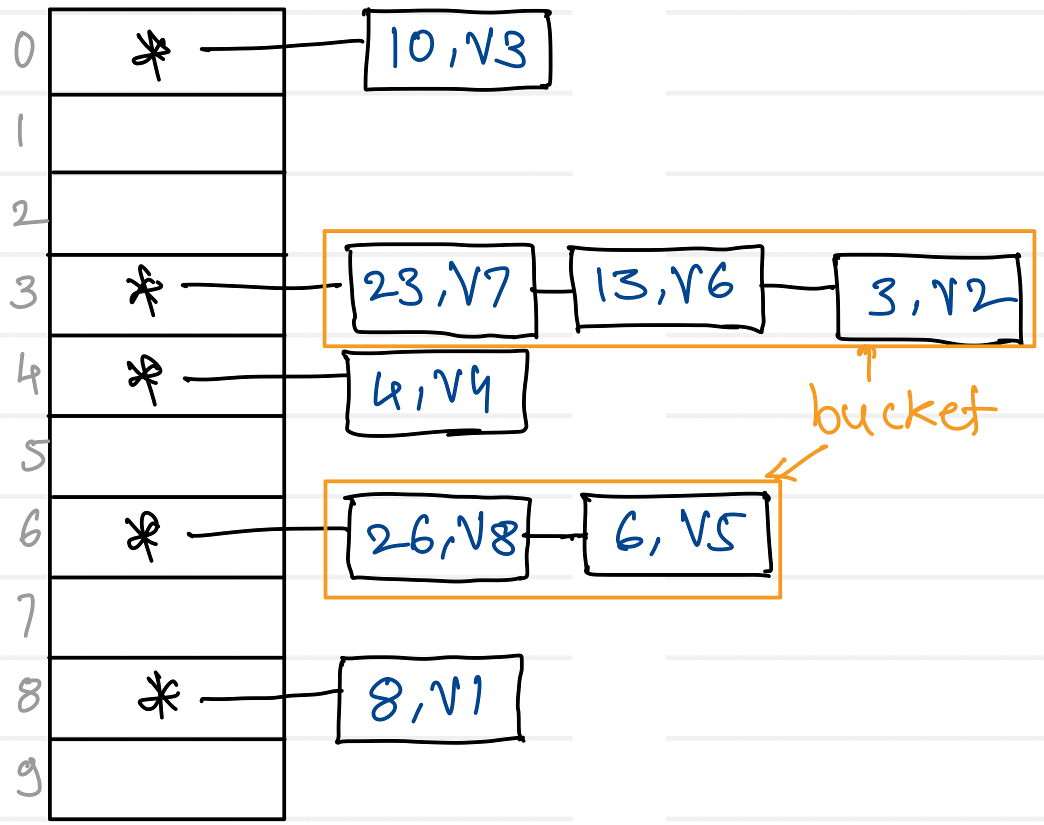
Search : O(1)
1. slot = h(k)
2. return arr[slot]

Delete : O(1)
1. slot = h(k)
2. arr[slot] = null

Closed Addressing / Chaining / Separate Chaining

8 - V1
3 - V2
10 - V3
4 - V4
6 - V5
13 - V6
23 - V7
26 - V8

size = 10



Hash Table

h(k) = k % size

h(13) = 13 % 10 = 3
h(23) = 23 % 10 = 3
h(26) = 26 % 10 = 6

Advantage :
- multiple key-value pairs can be stored into hash table

Disadvantages :
1) key value pairs are stored outside the table
2) space requirement is more due linked list.
3) Worst case time complexity is O(n) - when maximum keys will yield same slot.

Thank you!!!

Devendra Dhande

devendra.dhande@sunbeaminfo.com