

J	1	2	3	4	5	6	7	8	9	10	11	12	13
U	14	15	16	17	18	19	20	21	22	23	24	25	26
N	28	29	30										

21

TUESDAY

MAY

04

WK 19 • 124-241

8

A Graph is a collection of objects or entities called Nodes or Vertices connected through Edges.

Mathematically, a Graph G is an ordered pair of a set V of vertices and a set E of edges.

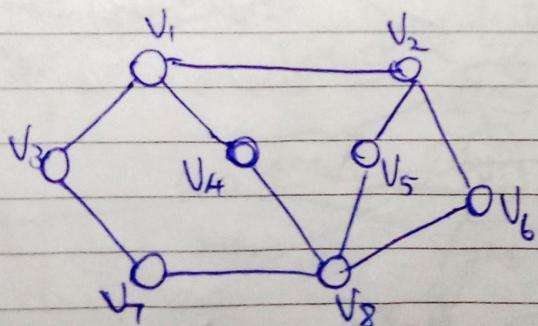
$$G = (V, E)$$

V = set of Vertices

E = set of Edges

ordered pair : $(a, b) \neq (b, a)$ if $a \neq b$

unordered pair : $\{a, b\} = \{b, a\}$ (origin, destination)



{ }

so set $V = \{V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8\}$

Directed

Undirected

2021

05

WEDNESDAY

MAY

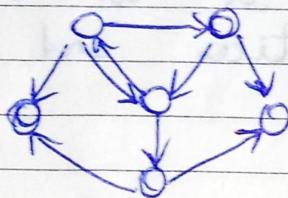
125-240 • WK 19

1	2	3	4	5	6	7	8	9	10	11			
12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30									

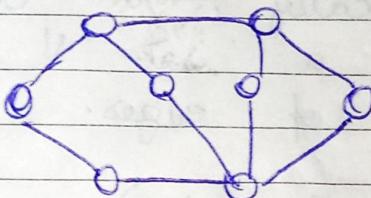
APR '21

$$E = \{ \{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_5\}, \{v_3, v_6\}, \\ \{v_4, v_5\}, \{v_4, v_8\}, \{v_5, v_6\}, \{v_5, v_7\}, \{v_6, v_7\}, \{v_7, v_8\} \}$$

Directed vs Undirected Graph

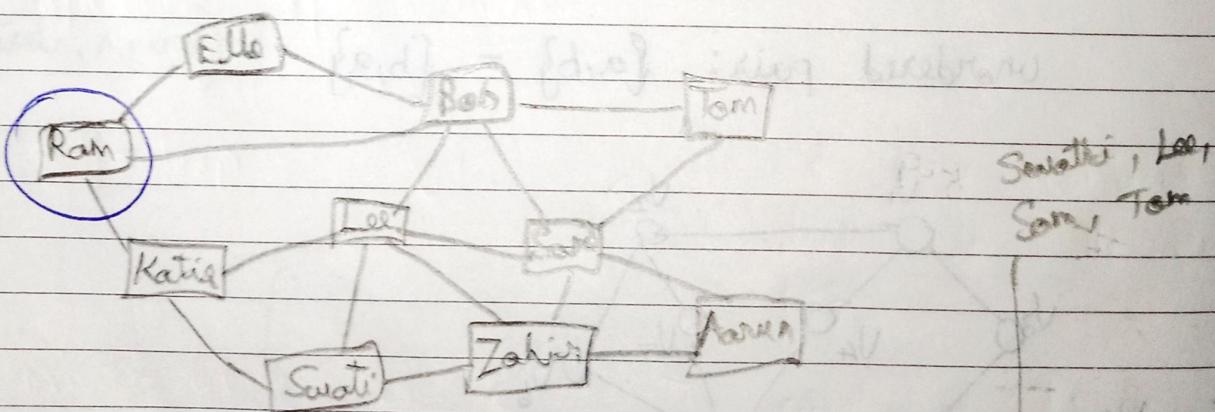


Directed



Undirected

ex) Social Network (facebook) - Undirected



Suggesting friends to Ram can be done by suggesting friends or Ram's friends.
(or)

2021

Shortest Path from Nadeel (Ram) equals 2

J	1	2	3	4	5	6	7	8	9	10	11	12	13	
U	14	15	16	17	18	19	20	21	22	23	24	25	26	27
N	28	29	30											
M	T	W	T	F	S	S	M	T	W	T	F	S	S	

21

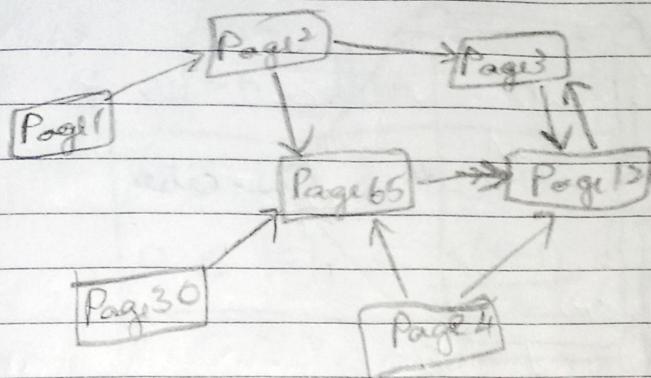
THURSDAY

MAY

06

WK 19 • 126-239

(Ex) Interlinked Web Pages - Directed Graph.

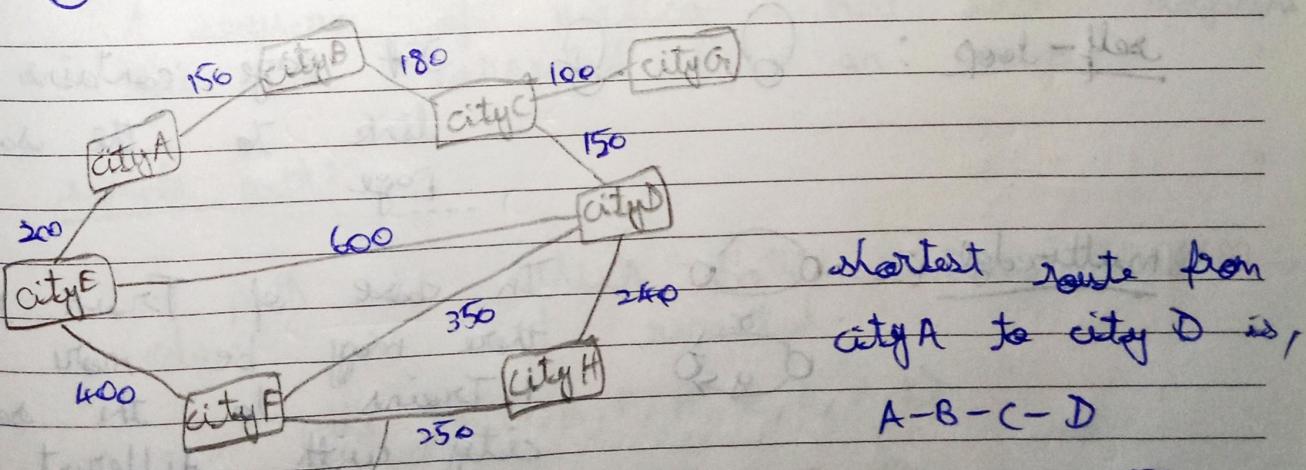


Page contains link
to another page

Web Crawling or Act of Visiting all nodes
in Graph

Weighted vs Unweighted Graph

(Ex) Intercity Road Network - Weighted



shortest route from
cityA to city D is,
A-B-C-D
 $\Rightarrow 150 + 150 + 150$

$\Rightarrow \textcircled{450}$

Undirected because of
both ways travel.

2021

07

FRIDAY

MAY

127-238 • WK 19

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
26	27	28	29	30						

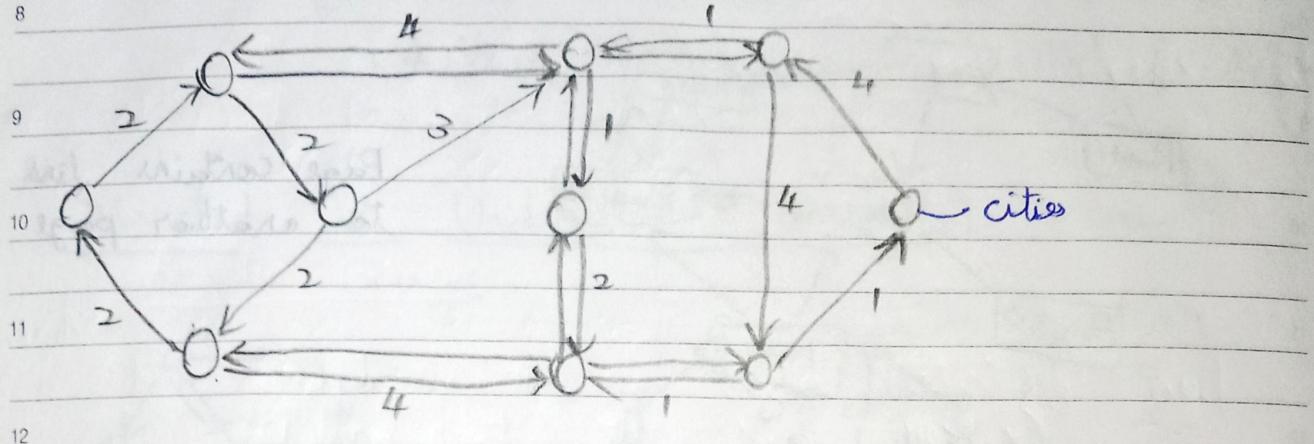
A

P

R

'21

ex) Intracity Road Network - Weighted Digraph.



In intracity road networks between ~~each node~~, each node two way links as well as one way is present. So a digraph is optimal.

Properties of Graphs

self-loop :

If page contains link to the same page.

multi-edge :

In case of Train routes there may be more than 1 trains to the same city with different costs, timings, etc.

J 1 2 3 4 5 6 7 8 9 10 11 12 13
 U 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 N 28 29 30
 M T W T F S S M T W T F S S

21

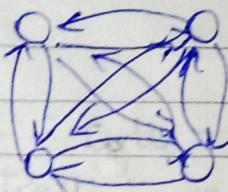
SATURDAY

MAY

WK 19 • 128-237

08

number of edges:



$$V = \{v_1, v_2, v_3, v_4\}$$

$$|V| = 4$$

if $|V|=n$, then

$$0 \leq |E| \leq n(n-1)$$

, if directed

$$0 \leq |E| \leq \frac{n(n-1)}{2}$$

, if undirected.

dense graph & sparse graph:

Too many edges

Too few edges

Adjacency Matrix Adjacency List to store the graph.

path:

A sequence of vertices where each adjacent pair is connected by an edge

simple path:

A path in which both vertices and edges are not repeated.

$$\text{path} = \langle A, B, F, H, E, B, A, D \rangle$$

$$\text{simple path} = \langle A, B, F, H \rangle$$

2021

10

MONDAY

MAY

130-235 • WK 20

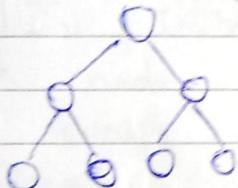
1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
26	27	28	29	30						

Closed walk

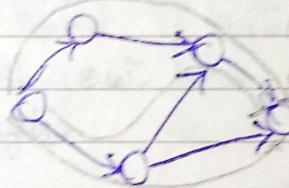
A walk that starts and ends at the same vertex.

Acyclic Graph

graph with no cycles (starting and ending at the same vertex)



Undirected
Acyclic graph



Directed
Acyclic Graph

(no cycles)

J 1 2 3 4 5 6 7 8 9 10 11 12 13
U 14 15 16 17 18 19 20 21 22 23 24 25 26 27
N 28 29 30

'21

TUESDAY

MAY

11

WK 20 • 131-234

Graph Representation

8

Edge List

9

~~Notes~~

10

Class Edge:

startVertex, endVertex, weight

11

Vertex List

Edge List

12

0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H

1

2

3

4

5

6

2021

12

WEDNESDAY

MAY

132-233 • WK 20

	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23
26	27	28	29	30							
M	T	W	T	F	S	S	M	T	W	T	F

APR

'21

Adjacency Matrix $O(V^2)$

Vertex List

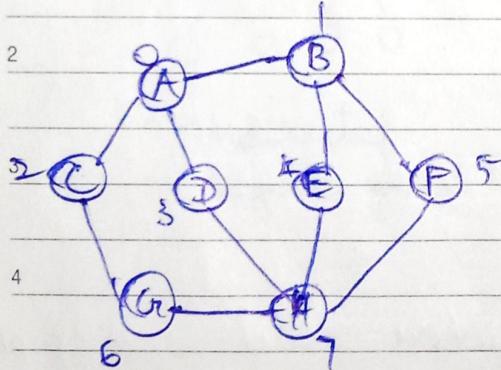
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H

Adjacency Matrix

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

values are stored even if nodes are not connected.

weights

Operation

finding adjacent nodes -

Time - Cost $O(V)$ Spec $O(n)$

finding if two nodes are connected

 $O(1) + O(n)$

J	1	2	3	4	5	6	7	8	9	10	11	12	13	
U	14	15	16	17	18	19	20	21	22	23	24	25	26	27
N	28	29	30											
'21	M	T	W	T	F	S	S	M	T	W	F	S	S	

THURSDAY

MAY

13

WK 20 • 133-232

But the space complexity is very high because it is not efficient in storage, even if there is no edge it has to store value '0'.

10

11 Adjacency List

12 Most real-world graphs are sparse, meaning
 1 most of the connections are "0." Our number
 1 of connections is small compared to total
 2 number of possible connections.

For a Social network with a billion (10^9)
 3 users,

4 row size = 10^9
 5 (1 user)

6 If the user has 1000 friends:

$$\text{no. of 1s} = 1000 \approx 1 \text{ KB} \quad 0, 1 \text{ are boolean}$$

$$\text{no. of 0s} = (10^9 - 1000) \approx 1 \text{ GB}$$

Adjacency Matrix

- find if $O(1)$
 2 nodes
 are
 connected
- adjacent nodes $O(1)$

Adjacency List

$O(V)$ — linear search
 $O(\log V)$ — binary search (BST)

 $O(N)$

2021

14

FRIDAY

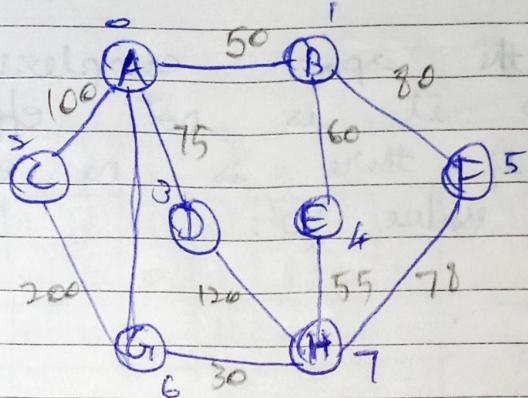
MAY

134-231 • WK 20

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
26	27	28	29	30						
M	T	W	T	F	S	S	M	T	W	F
R										

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

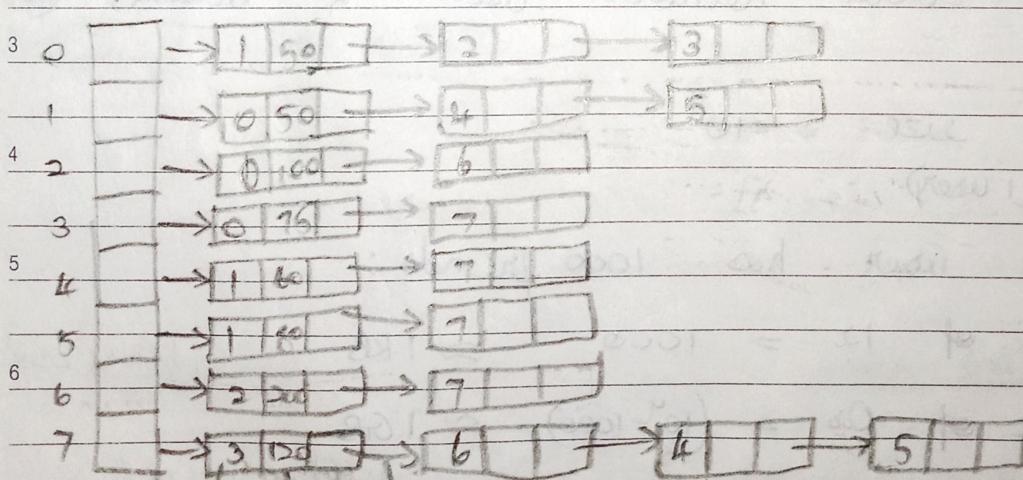
~~0 1 2 3 4 5 6 7 8 9 10 11~~



Instead we use an Array of linked lists.

Adjacency List

Undirected Graph



vertex

weight

edge

class

self-data = data

self-next = None

self-weight = ~~None~~ weight
(or)
None

J 1 2 3 4 5 6 7 8 9 10 11 12 13
 U 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 N 28 29 30
 M T W T F S S M T W T F S S

'21

SATURDAY

MAY

15

WK 20 • 135-230

How good is it if you want to add or delete a new connection in complexity?

(Consider using BST instead of LinkedList)

BFS

DFS

shortest distance

b/w nodes

finish Trees first - BST, BinaryTree
(Depth first traversal)

then finish Graphs - BFS, DFS, shortest distance.

then go to AVL trees, Tree, Red-Black Trees

top left or right or both,

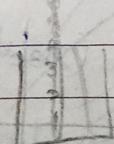
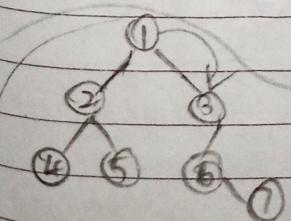
1, 2, 3, 4, 5, 6, 7

→ 7, 6, 5, 4, 3

Reverse Level Order

Traversal

SUNDAY 16



7 - 4 5 6 - 2 3 - 1

1 3 5 6 5 4 7

Queue

2021

17

MONDAY

MAY

137-228 • WK 21

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
26	27	28	29	30						

M T W T F S S M T W T F S S

'21

deletion in a binary tree

8

9

10

11

12

1

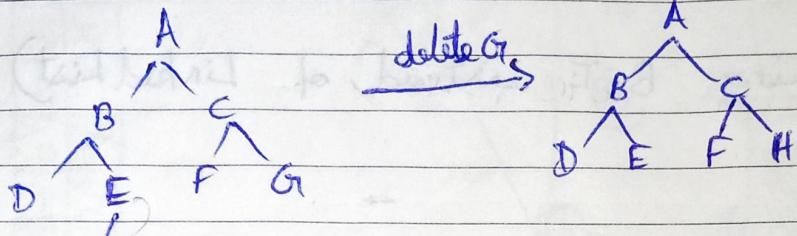
2

3

4

5

6



prenode = C

currnode = G

lastnode = H

prenode - right = lastnode

prenode = E

lastnode = H

prenode - left = None

~~delete (root, key)~~

delete (root, key)

delete (A, G)

{ delete (B, G)

delete (C, G)

~~if root == G:~~

if root.right == G:

root.right = None

qp = []
 qp.append (root)
 while all empty

return max (lheight, rheight)

J 1 2 3 4 5 6 7 8 9 10 11 12 13
 U 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 N 28 29 30

M	T	W	T	F	S	S	M	T	W	T	F	S
---	---	---	---	---	---	---	---	---	---	---	---	---

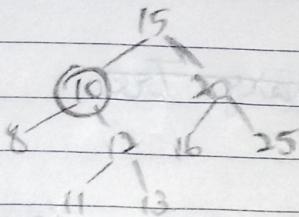
TUESDAY

MAY

18

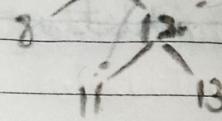
WK 21 • 138-227

21



rootNode = 10

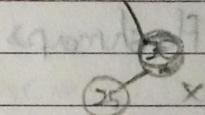
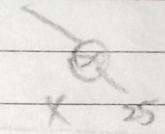
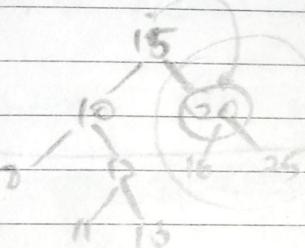
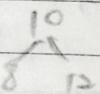
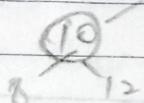
$SP = 10$
 $SC = 12$



rootNode = 10 $SP = 10$

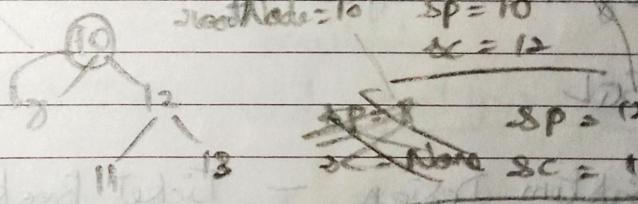
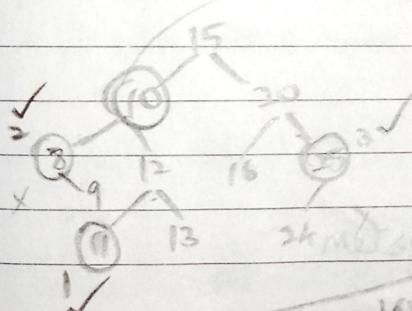
$SC = 12$

$SP = 12$
 $SC = \text{None}$

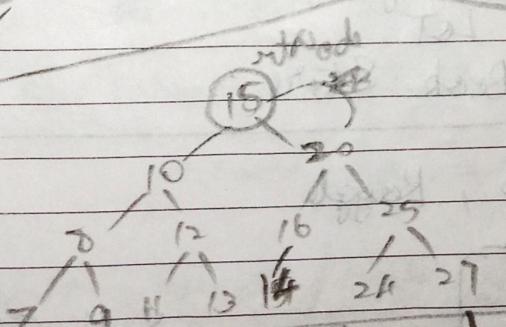


if rootNode.left is None

if rootNode.right is None



$SP = 12$
 $SC = \text{None} SC = 11$



$SP = 15$
 $SC = 20$

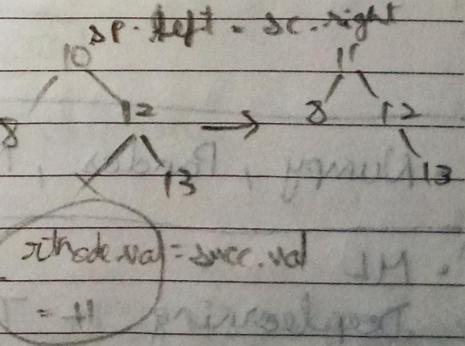
$SP = 20$
 $SC = 16$

~~SP = 16~~
~~SC = None~~

$SP = 16$
 $SC = 14$

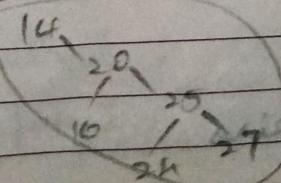
$16 + 15 = 15$

16



$rootNode = \text{None}, val = 14$

26A



existing tree about

electronic circuit (2) 2021

2021

24

MONDAY

MAY

144-221 • WK 22

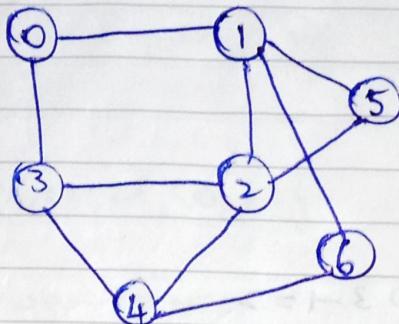
	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23
26	27	28	29	30							

M T W T F S S M T W T F S S

A P R

'21

Graph Traversals

BFS

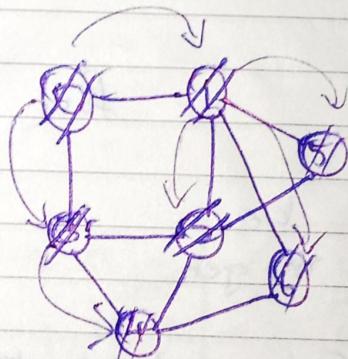
$$Q = \emptyset$$

$$Q = \emptyset \times \{3\}$$

$$Q = \emptyset \times \emptyset = \emptyset$$

$$Q = \emptyset \times \emptyset \times \emptyset \times \{4\}$$

0 1 3 2 5 6 4



- A node is inserted initially
- Immediate connections of nodes which are unvisited are added to the queue
- Upon deletion of a node the node is printed and its adjacents are added to queue.

J 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25 26 27
U 28 29 30
N M T W T F S S M T W T F S S

TUESDAY

MAY

25

WK 22 • 145-220

21

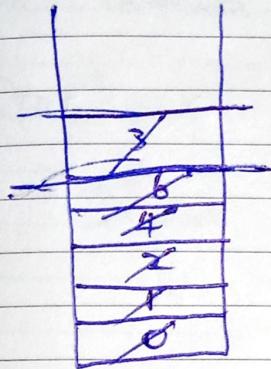
DFS

8

Stack

9

10



11

12

1

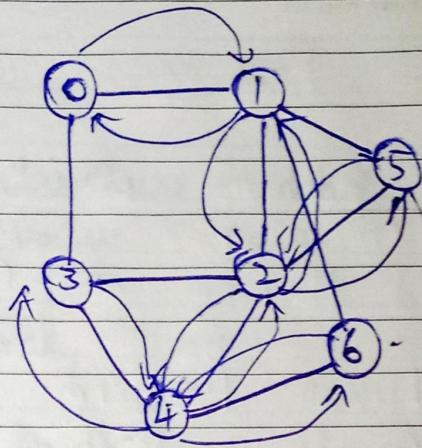
2

3

4

5

6



Print = 0 1 2 4 6 3 5

2021

J 1 2 3 4 5 6 7 8 9 10 11 12 13
U 14 15 16 17 18 19 20 21 22 23 24 25 26 27
N 28 29 30

THURSDAY
MAY

27

WK 22 • 147-218

Hashtables

A Hashtable is a structure where data is stored in key value pairs.

Operations performed are,

Insert , Search , Delete

O(1) O(1) O(1)

Representation

The key-value pairs are stored in an Array structure where the indexing is done by using a Hash function on the key.

$$\boxed{\text{Hash (key)} \rightarrow \text{index}}$$

A famous hash function is,

djb2

A hash function should be,

- fast to compute
- avoid collisions

28

FRIDAY

MAY

148-217 • WK 22

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
25	26	27	28	29	30					

M T W T F S S M T W T F S S

A P R

'21

Hashing

8

① Open Hashing or Separate Chaining

9

② Closed Hashing or Open Addressing

10

- Linear Probing
- Quadratic Probing
- Double Hashing

11

1 All the hashing techniques face collisions so there are methods to avoid this for each.

2

3



Separate Chaining

4

5

6

In the Hashtable when collisions occur that spot is chained by the incoming element like a linked list.

Ex A = 3, 2, 9, 6, 11, 13, 7, 12

$$h(k) = 2k + 3$$

$$\boxed{m=10}$$

use Division Method & Closed Addressing to store these values.

1 2 3 4 5 6 7 8 9 10 11 12 13
 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 28 29 30

N M T W T F S S M T W T F S S

21

8	
9	
10	
11	
12	
13	
14	6
15	11
16	
17	2
18	7
19	12
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	3
30	13

$$h(k) = 2k + 3 \quad m=10$$

$$\text{location} = (2k+3) \% 10$$

SATURDAY

MAY

29

WK 22 • 149-216

key	location
3	9
2	7
9	1
:	:
12	7

Linear Probing

Insert k_i at the first free location from $(i+k) \% m$ where $i=0$ to $(m-1)$.

Ex A = 3, 2, 9, 6, 11, 13, 7, 12 | $h(k) = 2k + 3$ | location = $(2k+3) \% 10$

0	13	<--
1	9	-
2	12	<--

② ② ③ ⑥

no of probes

order = 13, 9, 12, -, -, 6, 11, 2, 7, 3

SUNDAY 30

2021

31

MONDAY

MAY

151-214 • WK 23

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
26	27	28	29	30						

M T W T F S S M T W T F S S

A P R

'21

Quadratic Probing

8

Insert k_i at the first free location from
 $(u+i^2) \% m$ where $i = 0$ to $(m-1)$.

It is generally better than Linear Probing.

11

12 ~~Double Hashing~~

Double Hashing

1

Insert k_i at the first free location from
 $(u+V \cdot i) \% m$ where $i = 0$ to $(m-1)$.

3 ~~Ex~~ $A = 3, 2, 9, 6, 11, 13, 7, 12 \mid h_1(k) = 2k+3 \mid h_2(k) = 3k+1$

4 location = $(2k+3) \% 10$

$V = (3k+1) \% 10$

Division method

A

Double hashing

0	
1	9
2	
3	11
4	12
5	6
6	
7	2
8	
2020	3

$3 \rightarrow 9, 2 \rightarrow 7, 9 \rightarrow 1, \cancel{6 \rightarrow 5}, \cancel{12 \rightarrow 4}$

$11 \rightarrow \times$

$u = (u + V \cdot i) \% m = 5 + 8, 5 + 8 = 13$

$11 \rightarrow 3$

$13 \rightarrow \times$

$1 = 9 + 0, 9 + 0$

X not possible

$7 \rightarrow \times \quad \cancel{12 \rightarrow 4}$

J 1 2 3 4 5 6 7 8 9 10 11
U 12 13 14 15 16 17 18 19 20 21 22 23 24 25
L 26 27 28 29 30 31
M T W T F S S M T W T F S S

TUESDAY

JUNE

01

WK 23 • 152-213

21

order = -1, 9, -1, 11, 12, 6, -1, 2, -1, 3

8

9 Load Factor

10 Load factor = $\frac{\text{No of key-value pairs}}{\text{size of array}}$

11

12 Load factor must be lower. If its too low it may lead in memory inefficiency.

1

2

3

4

5

6

2021

19

WEDNESDAY

MAY

139-226 • WK 21

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
26	27	28	29	30						

A
P

'21

~~Finish trees first~~ — deletion in a Binary Tree

then finish — BFS, DFS, shortest distance (Dijkistra)
graph

then

Hashmaps

~~Ticks~~

Dynamic Programming

SQL

System Design — Ticket Booking System
Parking Lot
Online Book Store

• Numpy, Pandas, Matplotlib, Kaggle

• ML

Deep learning — TensorFlow

ML Ops

• AWS

• Docker and Containers

• CI/CD fundamentals, Jenkins

Soft Skills

Agile Methodologies

— Scrum

— Kanban

— Jira

2021

• Fast API, Tensorflow serving.