LECTURE-17

Priority Queue II

Priority Queue Operations and Analysis, Intro to Graph terminology.

For Poll Ev

CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan
Department of Computer Science, SBASSE

Credits: Some material taken from Dr. Ihsan Ayyub Qazi and Dr. Maryam Abdulghafur

Agenda

- Operations implementation in the PQ
- Analysis of PQ implemented as a Heap
- Introduction to Graphs

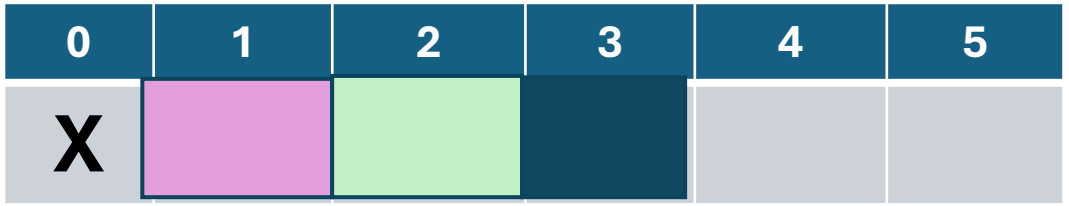

So Close! Only If you hadn't cut in line here!!

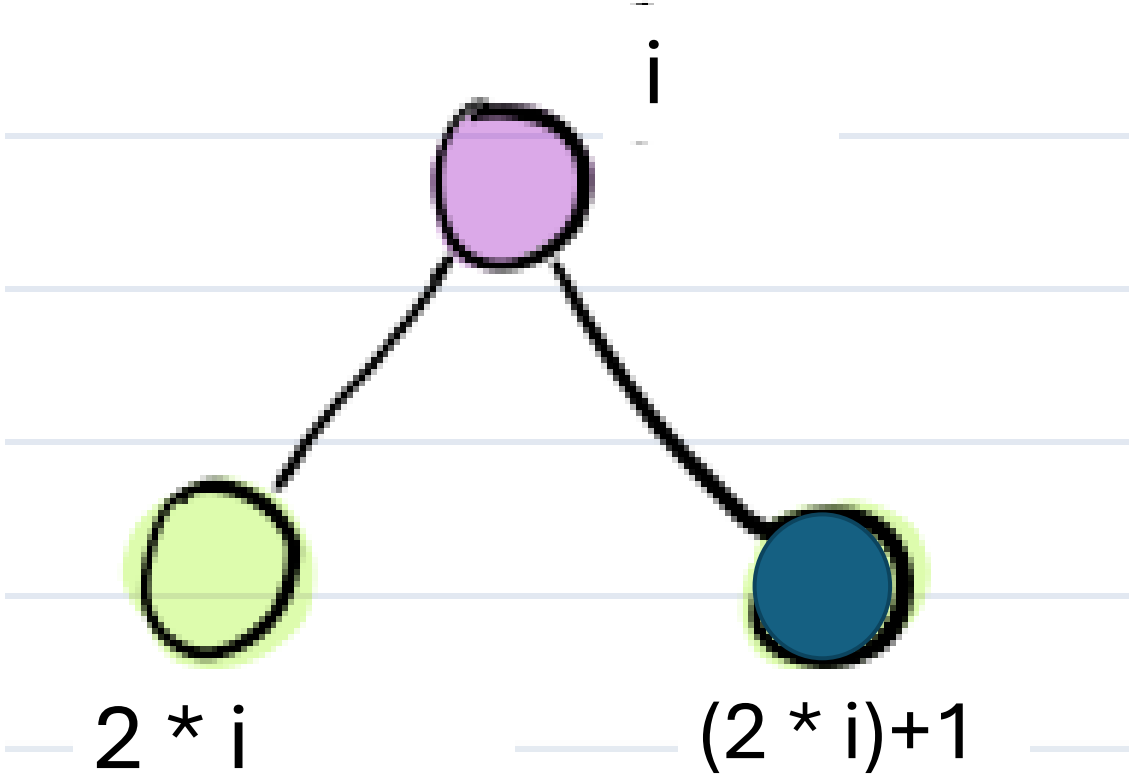
RECAP: Implement Priority Queue using a **binary heap**

Heap:

- Structural Invariant
- Value Invariant

Heap is stored in an array!





Finding Child/Parent Indices

Time for visualization! insert()

Q. If I add a data with key = 98 what index will it be added to in the array. After value field is fixed what index will it settle at?

0	1	2	3	4	5	6	7	8	9	10	11
X	45	39	29	20	10	28	17	7	5	8	

0	1	2	3	4	5	6	7	8	9	10	11
X	45	39	29	20	10	28	17	7	5	8	**98**

0	1	2	3	4	5	6	7	8	9	10	11
X	**98**	45	29	20	39	28	17	7	5	8	10

removeMax()

- Where is the element that will be deleted?

- How can we fill that hole?

Which value in any array can be removed most efficiently from the array?

removeMax()

How to remove the root which is the MAX key element?

1. Overwrite the value at the root with the last value in the array. Delete the last value in the array.

2. Compare the new value at the root with both its left child and right child, if a child or both is larger; swap it with the larger value of the two.

3. If no swap made then stop otherwise repeat 2 till you reach the leaf!

Time for visualization! removeMax()

Q. Which will be the first and last value to replace the root after removeMax() function is called once?

0	1	2	3	4	5	6	7	8	9	10	11
X	**98**	45	29	20	39	28	17	7	5	8	10

0	1	2	3	4	5	6	7	8	9	10	11
X	**10**	45	29	20	39	28	17	7	5	8	

0	1	2	3	4	5	6	7	8	9	10	11
X	45	39	29	20	**10**	28	17	7	5	8	

Priority Queue Operations Analysis

A PQ supports three basic operations just like a Queue ADT:

- **insert(e):** New element could be compared to and swapped all the way to the root of the tree! **O(logN)**
- **removeMin( ):** min element removed, its replacement at root could be compared to swapped all the way to the leaf! **O(logN)**
- **min( ):** just needs to return least/min value! **O(1)**

Which data structure other than an array does the performance match with closely!?

A Balanced BST!

How do **heap** and **balanced BST** compare?

Hints:

1. Which of these has the less complicated implementation?
2. How would you perform a findMinimum() in a **max heap**?

How to construct a heap? Activity!

Intuitive:

1. You are given N values to construct a heap out of.
2. Take a value from the N values and use heap insert() to add it to the heap.
3. Repeat 2 till you have used all N values.

Complexity: insert() of heap call N times

O(logN) * N = **O(logN)**

How to construct a heap? Floyd's method

- Add all unsorted values to an array.

- What is the condition for a heap?
  - Every node is root of a subtree, no child must have a larger value in the subtree!
  - If I trickle down the value to its right place and do the same for all other nodes

- A node will have to make how many swaps? As many as its height!

- In this case what will be total steps or cost of building a heap?

Complexity of Floyd's Build Heap

- 1/2 of the nodes are leaves → O(1)
- 1/4 of the nodes have at most 1 level to travel w/ percolateDown
- 1/8 of the nodes have at most 2 levels to travel w/ percolateDown
- …
- Work(n) = $\left(\frac{n}{2} * 1\right) + \left(\frac{n}{4} * 2\right) + \left(\frac{n}{8} * 3\right) + \ldots$

$= n(1/2 + 2/4 + 3/8 + \ldots)$

$= n(1/2\text{^}1 + 2/2\text{^}2 + 3/2\text{^}3 + \ldots)$

$= n\sum_{i=1}^{logn} \frac{i}{2^i} \leq n\sum_{i=0}^{\infty} \frac{i}{2^i} = 2n \in O(n)$

What happens if we repeatedly call removeMin() from a heap for all N items?!

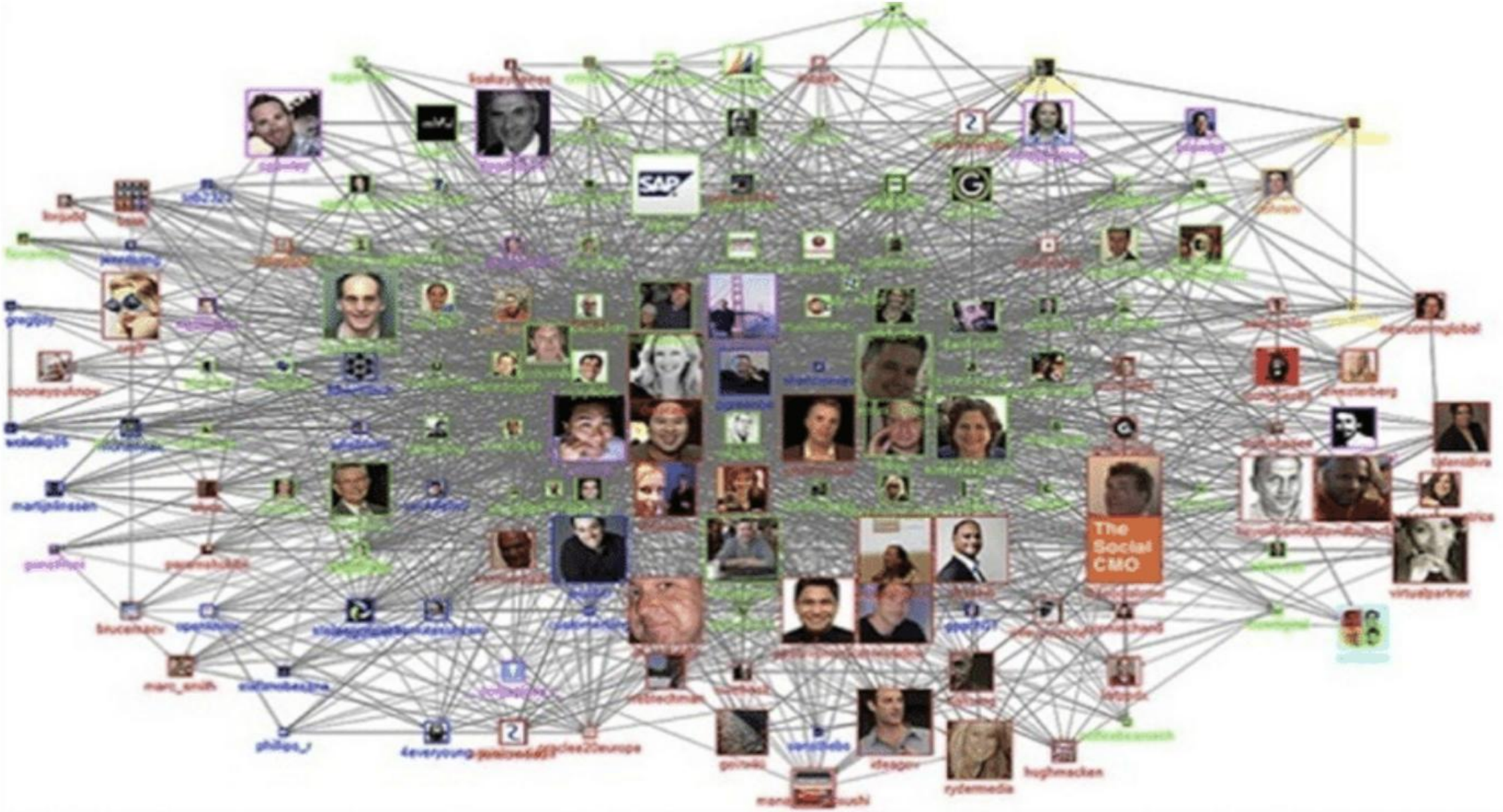
We will land with our N items in sorted order!

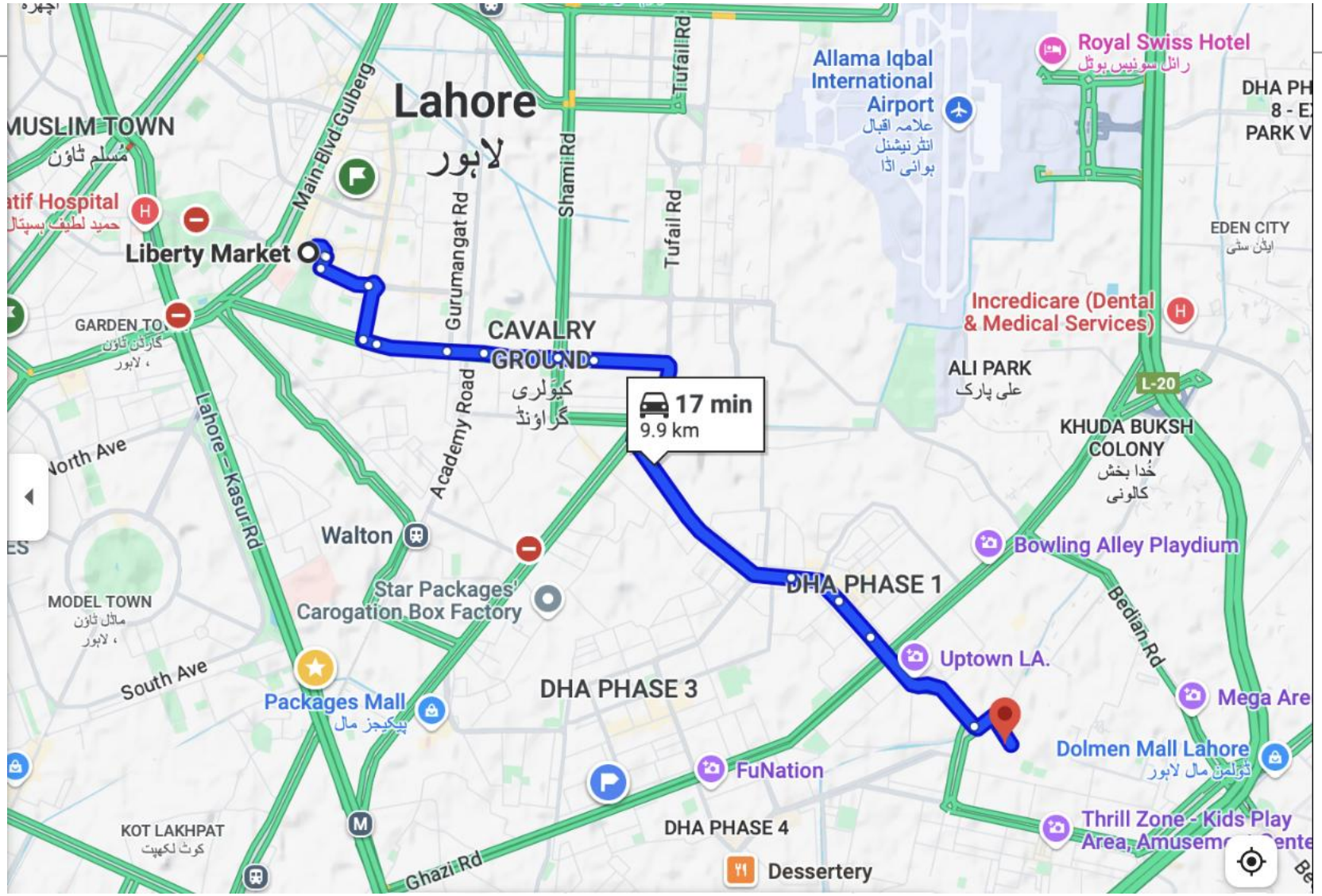
This algorithm is known as **Heap Sort**!

What is the complexity of HeapSort? **O(N * logN)**

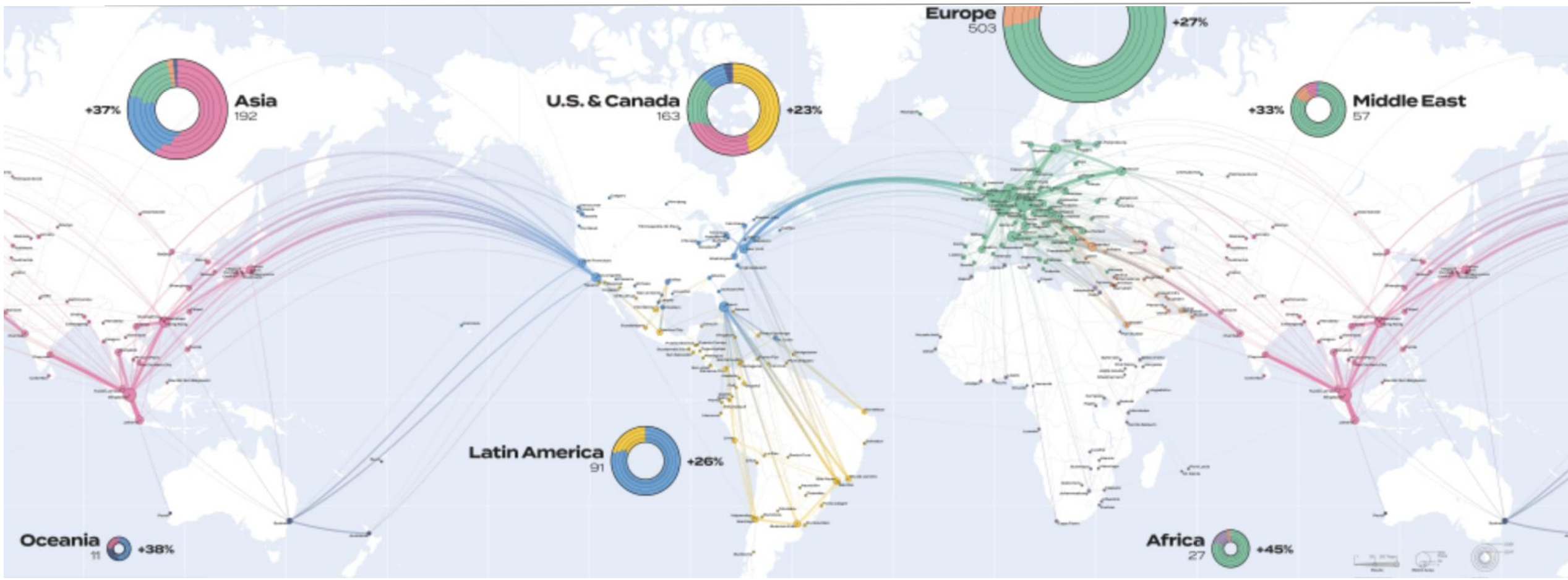
1. Build a heap **O(N)**
2. removeMin() for all elements **O(N * logN)**

Introducing Graphs! Activity!




Lahore
لاہور
Liberty Market
17 min
9.9 km
CAVALRY GROUND
کیولری گراؤنڈ
DHA PHASE 1
DHA PHASE 3
DHA PHASE 4
Walton
Star Packages' Carogation Box Factory
Packages Mall
پیکیجز مال
FuNation
Uptown LA.
Bowling Alley Playdium
Dolmen Mall Lahore
ڈولمن مال لاہور
Royal Swiss Hotel
رائل سوئیس ہوٹل
Allama Iqbal International Airport
علامہ اقبال انٹرنیشنل ہوائی اڈا
Incredicare (Dental & Medical Services)
ALI PARK
علی پارک
KHUDA BUKSH COLONY
خدا بخش کالونی
EDEN CITY
ایڈن سٹی
MUSLIM TOWN
مسلم ٹاؤن
MODEL TOWN
ماڈل ٹاؤن ، لاہور
KOT LAKHPAT
کوٹ لکھپت
Dessertery
Main Blvd Gulberg
Gurumangat Rd
Shami Rd
Tufail Rd
Academy Road
Lahore - Kasur Rd
Bedian Rd
Ghazi Rd
North Ave
South Ave
L-20

Internet Traffic Routes

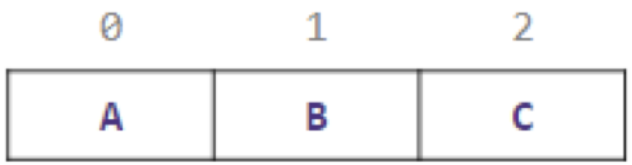


What is common in these images???

Inter-data Relationships

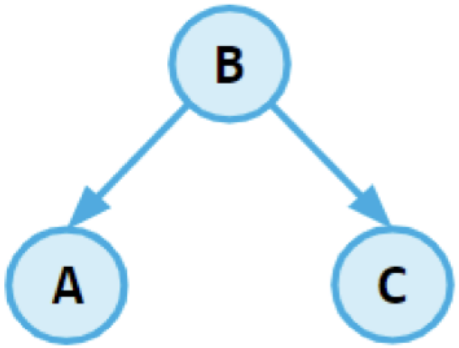
Arrays

- Elements only store pure data, no connection info
- Only relationship between data is order



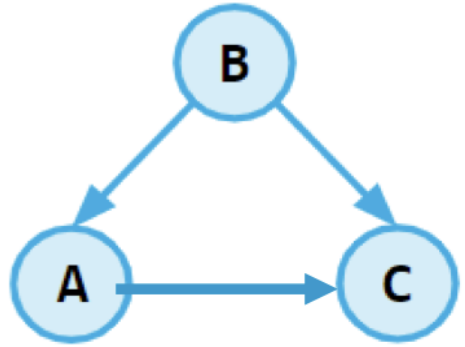
Trees

- Elements store data and connection info
- Directional relationships between nodes; limited connections



Graphs

- Elements and connections can store data
- Relationship dictate structure; huge freedom with connections

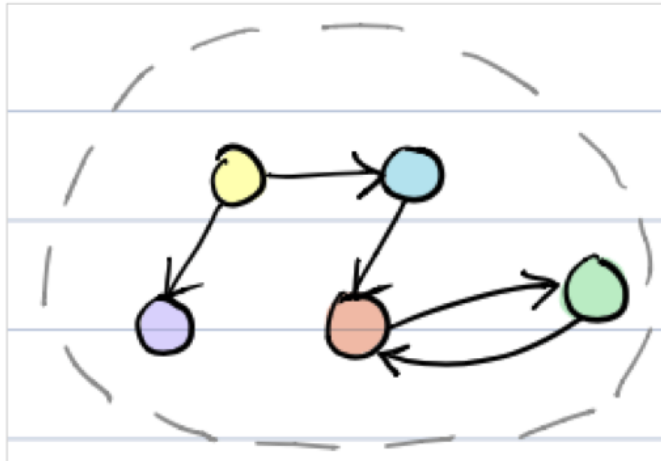


Graph Vocabulary

- Vertices / nodes
- Edges
- Directed vs undirected
- Weighted and unweighted

What is Graph?

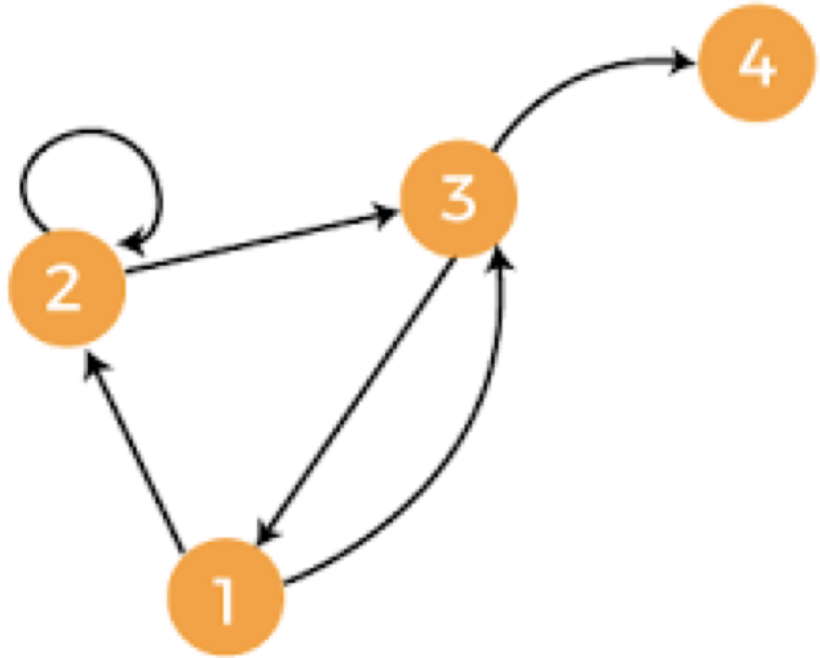
- A graph G consists of two sets, V and E → G=(V,E)
  - V: set of vertices (or nodes)
  - E: set of edges (pairs of vertices)
  - |V|: size of V, |E|: size of E



- Examples
  - Graph of webpages on the Internet (Web Graph)
    - V: webpages, E: hyperlinks between pages
  - Ways to walk between LUMS buildings
    - V: buildings, E: routes between buildings

Paths

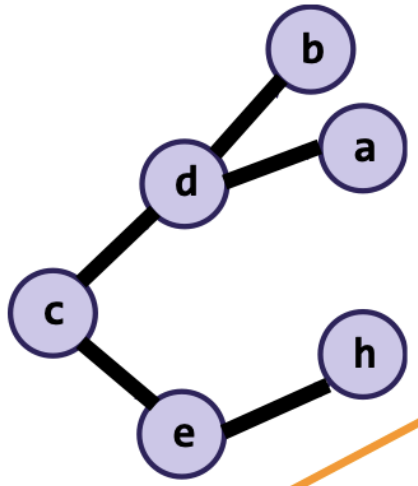
- **Path**: A path from vertex *a* to *b* is a sequence of vertices that connects a to b, represented as edges taken
  - A simple path repeats no vertices, except the first might also be the last
  - Example, one path from 1 to 4: {1, 2, 3, 4}



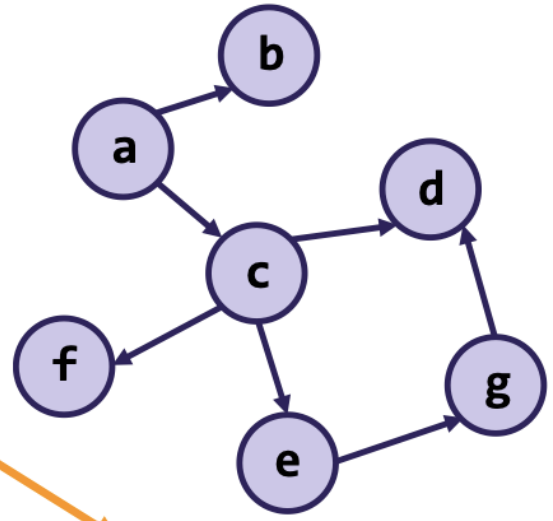
- **Path length**: Number of edges in the path
  - Distance is the length of the shortest path

- **Neighbor** or **adjacent**: Two vertices connected directly by an edge
  - Ex: 1 and 3

Graph Vocabulary



Graph Direction



Undirected Graphs

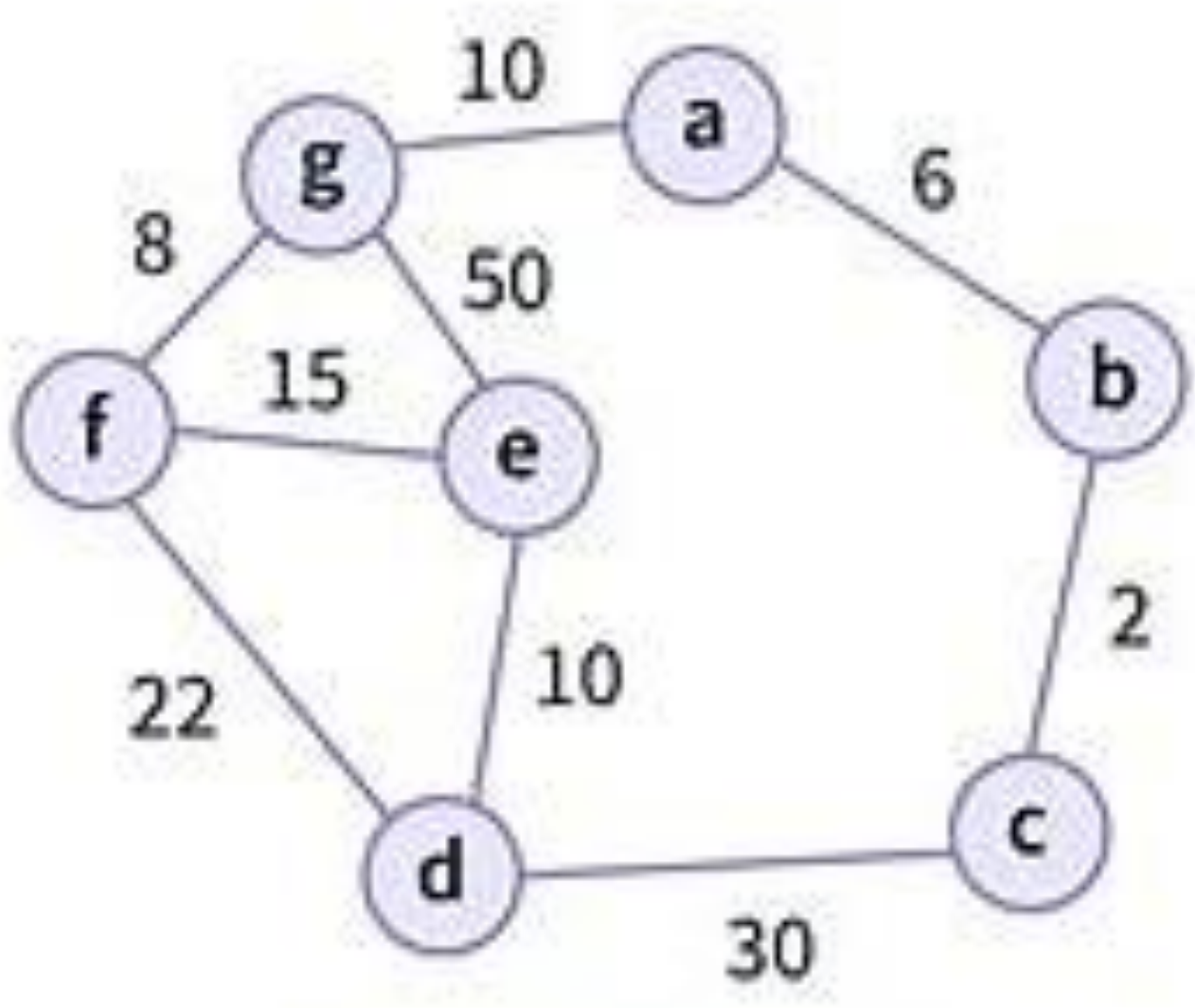
Directed graphs (Diagraphs)

Edges have no direction and are two way

E = {(e, c), (c, e), (d, b).....}

Edges have direction and are one way

E = {(a, b), (a, c), (c, f), (c, e), (c, d), (g, d)}



Weighted graph