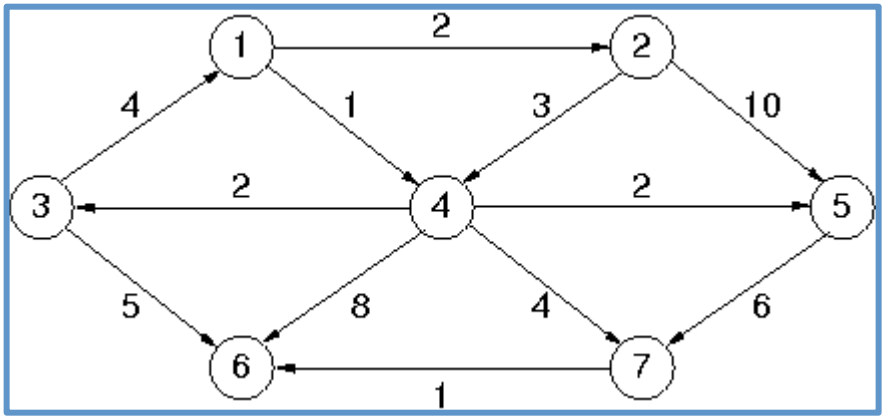
GRAPH ALGORITHMS: DOUBT-CLEARING SESSION



Partha P. Chakrabarti & Aritra Hazra

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

[Lecture – I]

Graphs-I: Introduction to Graphs

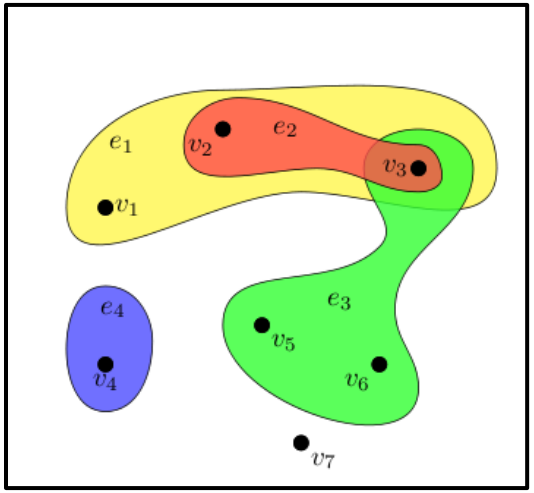
Participants: 141 / 159

(till 04-Apr-2020)

Graphs-I: Introduction to Graphs

Query: Please elaborate more on Hyper-edges (Slide-5)

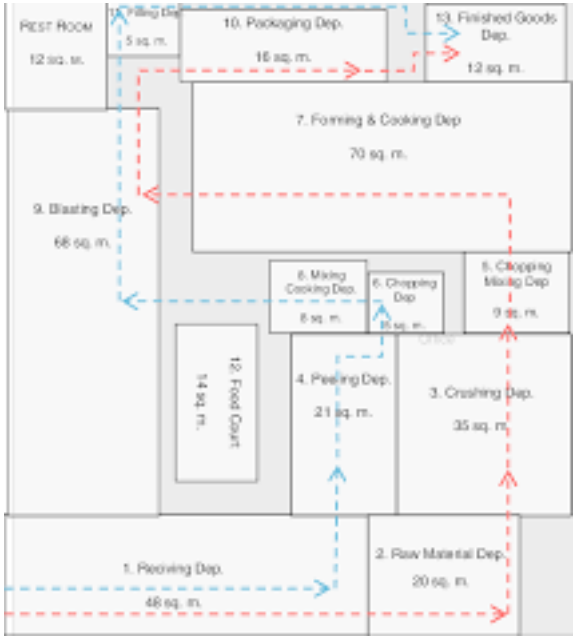
Answer:



Graphs-I: Introduction to Graphs

Query: Please elaborate more on Layouts (Slide-7)

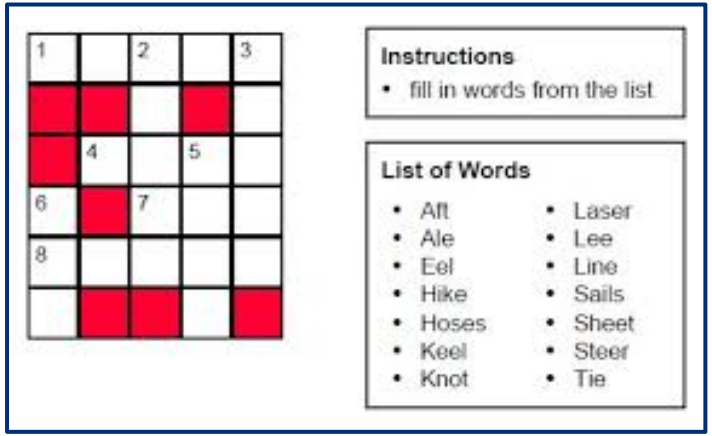
Answer:



Graphs-I: Introduction to Graphs

Query: When an edge is not labelled, do we take any value for the weight such as 1 or 0, or is it just considered as a relationship?

Answer: It is just a relationship or a connection whose meaning may be separately defined



Graphs-I: Introduction to Graphs

Query: Is Adjacency list representation is just a linked list? if so then for a node [1] having edges to both nodes [3] and [4], the representation 1→3→4 could also be made 1→4→3 ?

Answer: Yes. Its usage will depend on the algorithm

Graphs-I: Introduction to Graphs

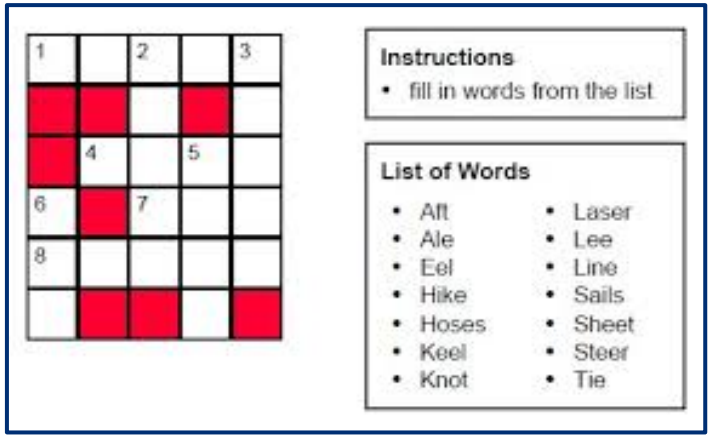
Query: Will there be any graph with both directed and undirected edges? (for ex., in circuits: resistors are undirected edges but voltage source and current source are directed edges)

Answer: Yes, it depends on the meaning of the specific type of edges.

Graphs-I: Introduction to Graphs

Query: During discussion of application of graphs (constraints) in Slide-9, how the edges in the graph are drawn based on constraints (ex., if 5-7 has a constraint, then 7-8 must not have one)?

Answer: No. The word related to 5 goes vertically down and that related to 7 goes horizontally . Thus their words intersect. On the other hand, for 7 and 8, both go horizontally and their words do not intersect.



Graphs-I: Introduction to Graphs

Query: While making a graph in a Data Structure, do we connect the nodes using pointers like in a multi-directional linked list, or we just simply store the node numbers in a array with additional information being which node numbers it is connected to?

Answer: There are many options. Both arrays and linked lists may be used. Linked lists can also be implemented in different ways including dynamic data structures or arrays.

[Lecture – II]

Graphs-II: Traversal of Undirected Graphs

Participants: 138 / 159

(till 04-Apr-2020)

Graphs-II: Traversal of Undirected Graphs

Query: In breadth first search (Slide-11), while we are enqueuing the adjacent nodes (succ(i)) into the queue, do we need to check whether it is present in the queue already?

Answer: Yes we do. We can do that by marking it enqueued. Otherwise we could also mark the node visited as soon as it enters the Queue.

```
Visited[i] all initialized to 0
Queue Q initially {}
 BFS_v2(s) {
  visited [s] = 1;  Q = {s};
  While Q != {} {
    j = DeQueue (Q);
    For each k in succ (j) {
      If (visited k] == 0) {
                visited[k] = 1;
                EnQueue(Q,k);
            }   }
    }
```

Graphs-II: Traversal of Undirected Graphs

Query: In path-finding algorithm (Slides 11+12), shouldn't cost [k] be initialized to infinity, instead of zero? Also, elaborate how this finds shortest cost path in weighted undirected graph.

Answer: Yes. These are corrected in the updated slide in a later lecture that was circulated.

```
G = (V,E) / Assume positive edge costs/
visited[i] all initialized to 0
cost[j] cost from s to j, all initialized to ∞
Ordered Queue OrQ initially {}
 BFSW(s,g) {
 cost [s] = 0; OrQ = {s};
 While OrQ != NULL {
   j = Remove_Min (OrQ); visited[j] = 1;
   if (j == g) terminate with solution cost[j];
   For each k in succ (j) {
   If (visited[k] == 0) {
            if (cost[k] > (cost[j] + C[j,k])) {
                        cost[k] = cost[j] + C[j,k];
                        Insert_Reorder(OrQ,k);}
              }
   } }
 If OrQ is empty terminate ("No Solution");
} / This method is  called Dijkstra's Algorithm /
```

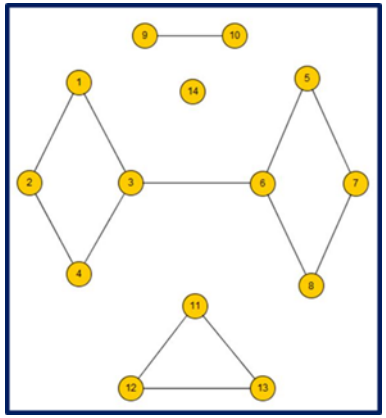
Graphs-II: Traversal of Undirected Graphs

Query: In Slide-10, why the concept of time stamp is required? Why this is important – didn't understand the conceptual problem.

Answer: This was discussed in details in Lecture 3 and is specially useful for directed graphs.

```
Visited[i], comp[i] all initialized to 0
count = 0;
Algorithm components() {
for each node k do {
  if visited [k] == 0 { count = count + 1;
    DfComp_S(k) }

DfComp(node) {
   visited[node] = 1; comp[node] = count;
   for each j in succ(node) do {
      if (visited [j] ==0) DfComp(j)
              }
 }
```



Graphs-II: Traversal of Undirected Graphs

Query: Path-finding algorithm overall is unclear as there will be many paths in a dense graph. Also, it is unclear how a path between any two nodes can be found out by parent pointer links.

Answer: The parent pointer marks one parent depending on the algorithm and has different usages. In DFS it marks the “Tree Edge” parent. In BFS it marks the earliest parent or the one whose path from start is shortest. The number of paths may be many.

Graphs-II: Traversal of Undirected Graphs

Query: Can you please provide the algorithm for path-finding using DFS? In path-finding using DFS, do we have to traverse and make parent links everytime for different pair of nodes?

Answer: Traverse back through the parent pointers.

Graphs-II: Traversal of Undirected Graphs

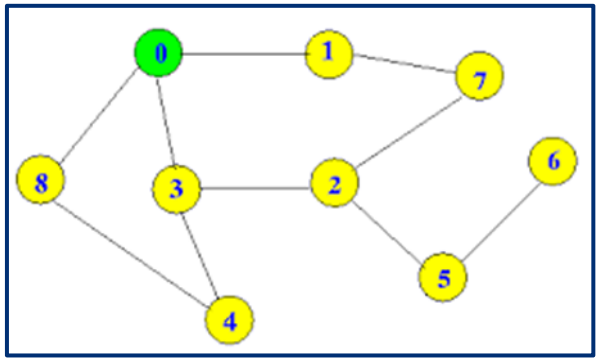
Query: What is the necessity of Bread First Search over Depth First Search ? Or, what are the differences between both? Can BFS be also modified to do other things as done with DFS?

Answer: We discussed the Shortest Path Problem. Yes, BFS can me modified as we have seen in Shortest Cost Paths using Best First Search.

Graphs-II: Traversal of Undirected Graphs

Query: In Slide-8, Like 4-8 link is not a back edge but 8-0 is, because 0 is from where we had started? So, that means if back edge exists, does the graph have a cycle? Please elaborate the cycle finding algorithm in details.

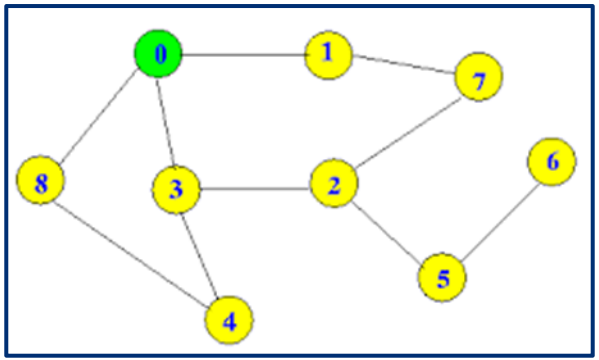
Answer: Yes because 0 was “visited”. Back edges which reach already visited nodes during DFS help to detect cycles.

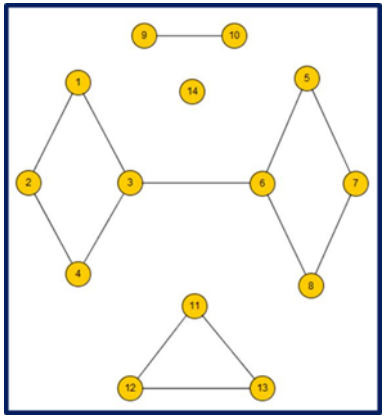


Graphs-II: Traversal of Undirected Graphs

Query: Can DFS be done iteratively? How the iterative version of DFS is implemented?

Answer: If it is a single connected component, then no. Otherwise Yes.





Graphs-II: Traversal of Undirected Graphs

Query: In Slide-11, how would we find the depth/level of node we are accessing right now while doing BFS? This would be necessary in shortest path finding as well, as this would help return the length of this path.

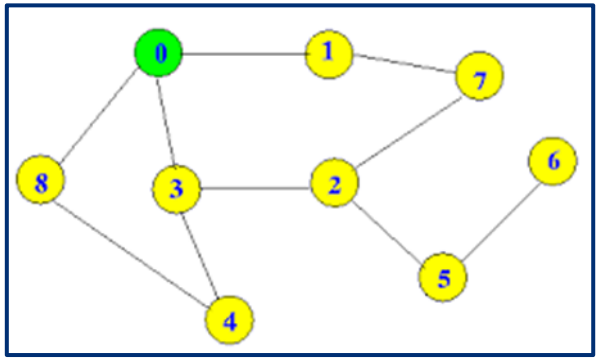
Answer: We discussed it in Lecture 5.

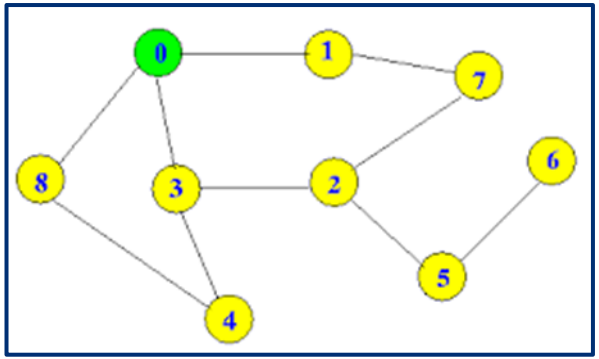
```
visited[i] all initialized to 0;
Length[i] length from s to i, all initialized to 0;
Parent[i] = parent of i  / initially Null/
Queue Q initially {}
 BFS(s, g) {
  visited [s] = 0;  Q = {s};
  While Q != NULL {
    n = DeQueue (Q);
    if (n== g) return with path through parent links;
    visited [n] = 1;
    For each k in succ (n)
          if (visited[k]==0) && (k is not already in Q) {
                    parent[k] = n;
                    Length[k] = Length[n]+1;
                    EnQueue(Q,k); }
      }
     If Q is empty then return with failure ("No Path");
}
```

Graphs-II: Traversal of Undirected Graphs

Query: Please explain how do we get O(|E| + |V|) for DFS and BFS?

Answer: Both these algorithms visit every node and every edge only once.





[Lecture – III]

Graphs-III: Traversal of Directed Graphs

Participants: 134 / 159

(till 04-Apr-2020)

Graphs-III: Traversal of Directed Graphs

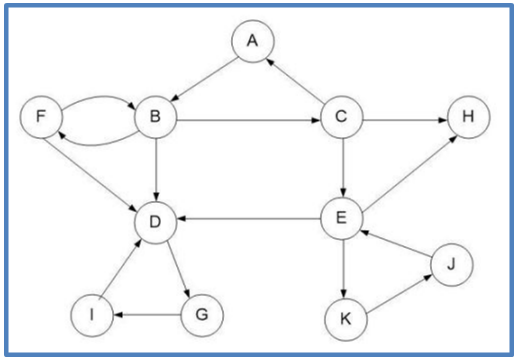
Query: Cross edge understanding is unclear.

– In the lecture a cross-edge was described as an edge where entries and exits overlap, but the condition given in Slide-6 doesn't match that.

– Also, it is mentioned that if a node is reached by another node then it passes through forward or tree or cross-edges. But if the nodes are related with cross edges then it means they are in different components – so that node can't be reached by that node.

Answer: There was a mistake in that spoken sentence. The condition is correct. The definition of component is in an undirected graph. For a directed graph it is a little different.

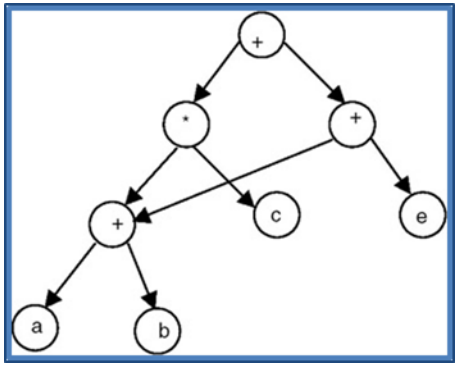
Edge (u,v) is
Tree Edge or Forward Edge: if & only if
Entry[u] < Entry[v] < Exit[v] < Exit[u]
Back Edge: if & only if
Entry[v] < Entry [u] < Exit [u] < Exit [v]
Cross Edge: if & only if
Entry [v] < Exit [v] < Entry [u] < Exit [u]



Graphs-III: Traversal of Directed Graphs

Query: Where is topological ordering used? What is exactly this topological ordering do?

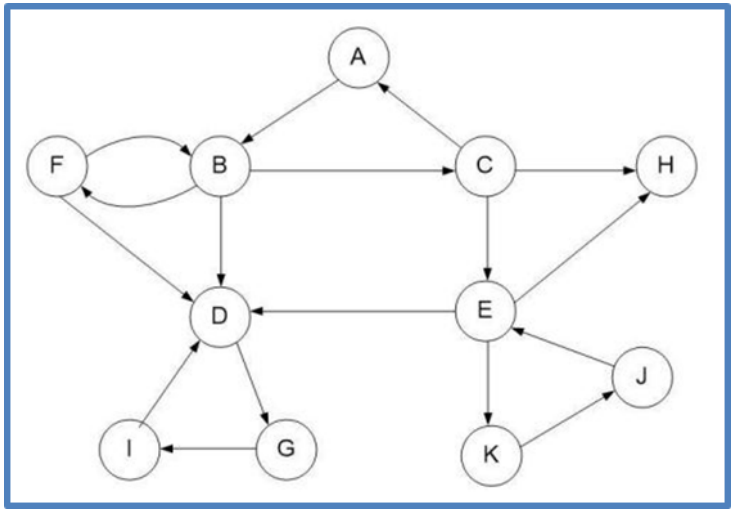
Answer: It is used to sequence items that may have precedence constraints



Graphs-III: Traversal of Directed Graphs

Query: Usually, DFS needs to be called multiple times to cover the whole graph. The choice of node matters in the coverage of each traversal (refer to Slide-4). So, is there any efficient way to find a node such that the traversal coverage gets maximized for a graph?

Answer: Yes, but that is not necessary. Doing that will require pre-processing and only increase the complexity.



[Lecture – IV]

Graphs-IV: Minimum Spanning Trees

Participants: 117 / 159

(till 04-Apr-2020)

Graphs-IV: Minimum Spanning Trees

Query: Can MST be created alternatively as: Initially go to any node, mark it visited and then among the unvisited successors of that node visit that node that is connected through a minimum edge weight. If there is no successors then back-track .

Answer: No, it may not work always.

Graphs-IV: Minimum Spanning Trees

Query: Some parts of Kruskal's algorithm is unclear (Slide-12)

– How cycles are being detected?

– The condition is while E is not empty or V_T=V . Shouldn't it be while E is not empty and V_T is not equal to V? – because, we terminate only when V_T= V or E is empty. YES, You are Correct

Answer: By Using the Disjoint Set Union-Find Data Structure. A separate lecture is there on this data structure.

```
Kruskal (V,E) {
Sort the edges in E in increasing cost;
VT = {}; ET = {}; cost = 0;
While E is not empty or VT != V do {
Choose the next minimum cost edge e = (n,m) in E;
E = E – {e}
If adding n, m in VT and e in ET makes a cycle,  discard e,
else {
      VT = VT + {n} + {m}
      ET = ET + { e}
      cost = cost + C[n,m] }
   }
Return <GT = (VT, ET), cost>
}
```

```
algorithm Kruskal_UF (G = (V,E)) {
A  = {};
for each v in V do MAKE-SET(v);
for each edge e = (u, v) in E ordered by increasing
weight(u, v) do
   {
    if FIND-SET(u) ≠ FIND-SET(v) then
    {
      A  = A + {(u, v)}
      UNION(FIND-SET(u), FIND-SET(v))
     }
    }
return A
}
```

Graphs-IV: Minimum Spanning Trees

Query: In Kruskal's algorithm, what does the FIND-SET operation mean? If we do not use hashing, then would this FIND-SET operation lead to increase in complexity?
How does the time complexity is O(|E|log|E|), not O(|V|log|E|) as we need to find (|V|-1) edges?

Answer: We shall discuss this in details in the Lecture on Disjoint Union Find Data Structures.

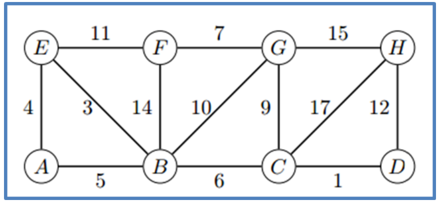
The Complexity is O(|E|log|E|) for Prim's because in the worst case every edge has to be inserted into the MinHeap and in Kruskal's Algorithm, we need to sort all the edges. Note that O(|E|log|E|) is the same as O(|E|log|V|)

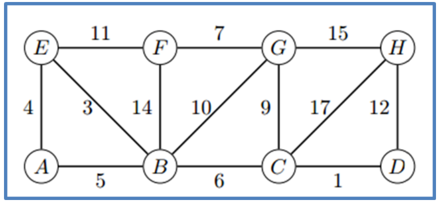
Graphs-IV: Minimum Spanning Trees

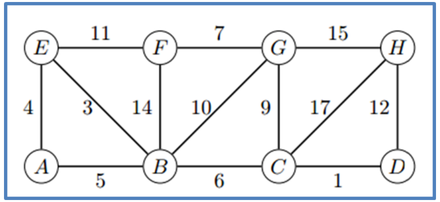
Query: Need more information about the values returned by the function MST. In Slide-5, what does cost' and T' symbolize?

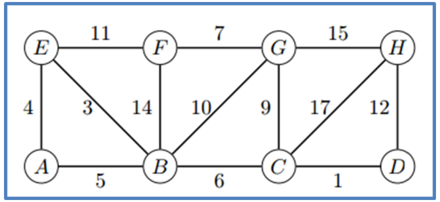
Answer:

```
MST(G1, G2,T) {
If (G2 =={}) return <T,0>
For each edge e = (n,m) from G1 to G2, recursively find the
minimum cost Tree (cost_a) using the remaining part of G2 and the
corresponding T' on selection of the edge e as a part of the MST
  {   G1a = G1 + {m};
      G2a = G2 – {m}
      Ta = T + {e}
      <T', cost'> = MST(G1a, G2a,Ta)
       cost_a =  C[n,m] + cost'
   }
Let <T_min, Cost_min> be the minimum cost_a and corresponding
T' found across all edges e
Return <T_min, Cost_Min>
}
```









[Lecture – V]

Graphs-V: Shortest Path in Graphs

Participants: 90 / 159

(till 04-Apr-2020)

Graphs-V: Shortest Path in Graphs

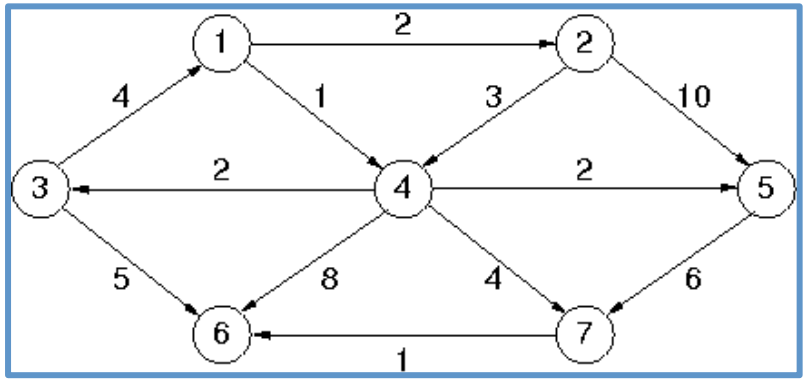
Query: In Slide-4, how to check that ‘k is not in queue’ – checking an element exists in queue itself may take O(n)? Please explain the necessity of this condition.

Answer: Use an array inQ[] where each node is marked inQ[i] or not. It can be updated and checked in constant time.

Graphs-V: Shortest Path in Graphs

Query: In Slide-5, why the algo will not work for directed cyclic graphs (video time: 25:24)? If we use a visited array, we are not going to revisit a node – so the algo should work .. please explain.

Answer: In a directed graph, visiting a node again by DFS does not mean that there is a cycle. For example, node 4 may be marked visited through the edge (1,4) and when you come back to it from (2,4) there is no cycle. So you need to check it in more details. Also in DFS you will come back to a node with smaller path cost from start and may need to redo. I gave it as an exercise.



[Lecture – VI]

Graphs-VI: All-pairs Shortest Paths in Graphs

Participants: 56 / 159

(till 04-Apr-2020)

Graphs-VI: All-pair Shortest Paths in Graphs

Query:

Answer:

Thank you