CS6033 Lecture 7 Slides/Notes

Elementary Graph Algorithms (Notes, Ch 20)

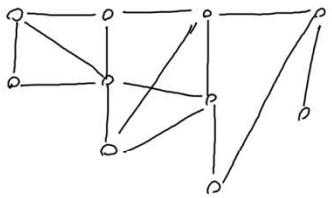
By Prof. Yi-Jen Chiang
CSE Dept., Tandon School of Engineering
New York University

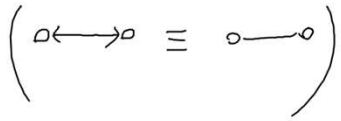
1

Elementary Graph Algorithms.

Graph: $G = (V, E)$ V: set of vertices, E: set of edges.

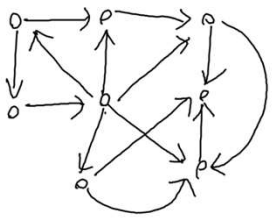
Undirected graph:





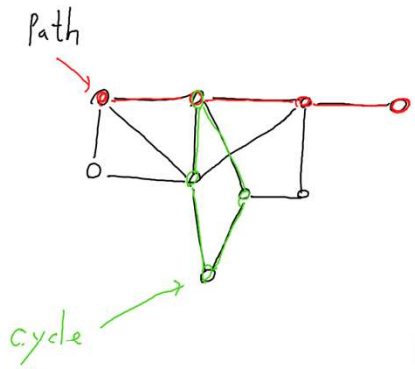
weighted graphs

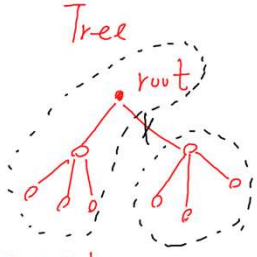
directed graph:



unweighted graphs

2





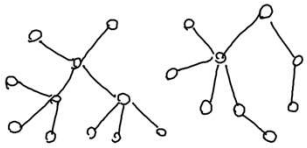
n vertices

n-1 edges.

A tree:

1. has the max. # edges without forming a cycle.
2. has the min # of edges to make the graph connected.

Forest



3

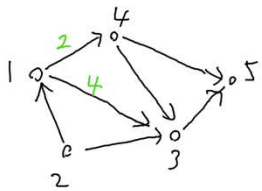
Representations of Graphs: $G = (V, E)$ $|E| \le \binom{V}{2} = \frac{V(V-1)}{2} = O(V^2)$

1. Adjacency List. eg. $E \sim V$

   $\sim V\sqrt{V}$.

2. Adjacency Matrix

Directed graph:

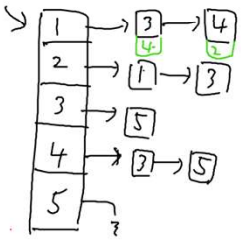


Query: $(1,5) \in E$?

$A_{15} = 1$? $O(1)$ time

weighted graph.

Adjacency List:



Space: $O(V+E)$

Adjacency Matrix: (A)

	1	2	3	4	5
1	0	0	1	1	0
2	1	0	1	0	0
3	0	0	0	0	1
4	0	0	1	0	1
5	0	0	0	0	0

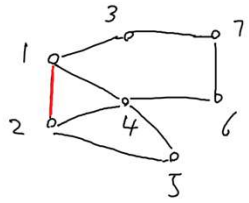
Space: $O(V^2)$

$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{else.} \end{cases}$

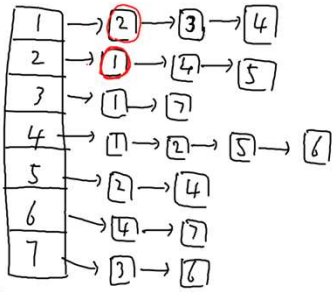
$A_{ij} = \begin{cases} w(i,j) & \text{if } (i,j) \in E. \\ 0 & \text{else.} \end{cases}$

4

Undirected graph



A. List (space: $O(V+E)$)



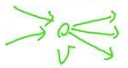
Each edge is represented twice

Total size of the edge lists is $2|E|$.

= Total degree of all vertices

Def: The degree of a vertex v. $deg(v)$ is the total # edges incident on v

(In directed graph. in-degree / out-degree of v.



in-deg(v) = 2
out-deg(v) = 3.)

A. Matrix (space: $O(V^2)$)

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

$(i,j) \in E$.

$A_{ij} = 1 = A_{ji}$

Symmetric matrix.

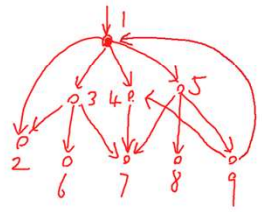
$A^T = A$ (A^T: transpose of A. $(A^T)_{ij} = A_{ji}$)

5

2 Main Graph Traversal Methods

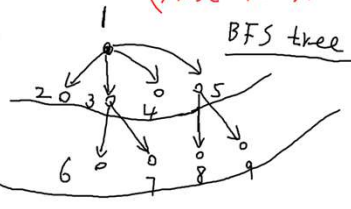
1. Breadth-First Search (BFS)
2. Depth-First Search (DFS)

BFS:



Use a FIFO queue (First-in-first-out)

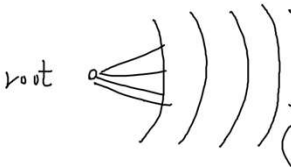
Q



BFS tree

black edges: tree edges.

Run time: $O(V+E)$



(wave-front propagation)

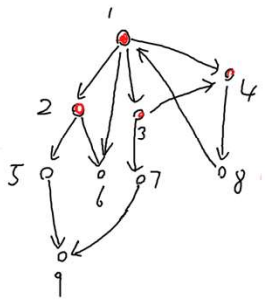
6

Depth-First Search

Each vertex is colored: { white: not yet visited.
gray: visited but not yet finished
black: finished.

Use a stack.

Label each vertex with { discovery time: time first visited
finish time: time the vertex is finished.



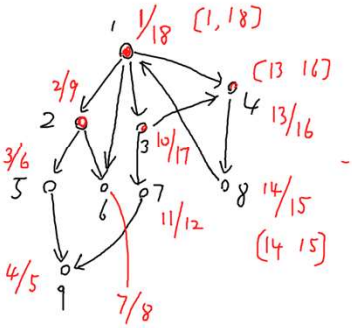
7

Depth-First Search

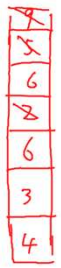
Each vertex is colored: { white: not yet visited.
gray: visited but not yet finished
black: finished.

Use a stack.

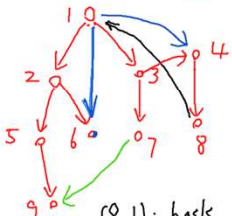
Label each vertex with { discovery time: time first visited
finish time: time the vertex is finished.



stack



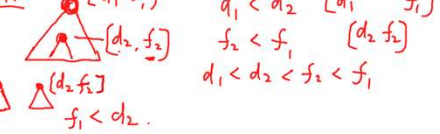
DFS-tree: red edges are the tree edges of DFS-tree.



(8,1): back edge.
(1,6), (1,4): forward edges.
(7,9): cross edge.

Lemma: All intervals are either disjoint or nested.

(disjoint: [ ] [ ]
nested: [[ ]]



* Each vertex has discovery & finish times.
forming an interval [discovery, finish]

8

Classification of Edges by DFS

1. Tree edge : $(u,v) \in$ DFS tree. v is a child of u in DFS tree.
   v is white when (u,v) is explored.
2. Back edge : v is an ancestor of u.  v is gray when (u,v) is explored.
3. Forward edge : v is a decendent of u but not a child of u 
   $\Rightarrow$ v is black when (u,v) is explored.
4. Cross edge : v is in a different subtree from u. and v is visited before u
   $\Rightarrow$ v is black when (u,v) is explored. 

cross edge? 

(v,u) is tree edge

For undirected graphs:
Claim: There are only tree edges and back edges

Pf:
Forward edge? 

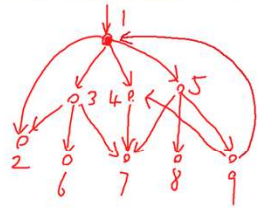
$(v,u)=(u,v)$ is back edge

$\Rightarrow$ There is NO forward edge or cross edge.

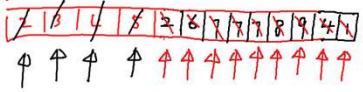
9

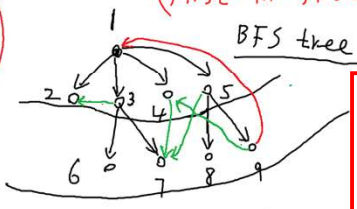
Similarly, we can classify edges by BFS

Eg.

BFS : 

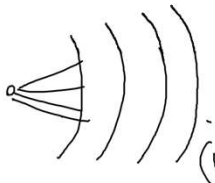
Use a FIFO queue
(First-in-first-out)

Q 



BFS tree

tree edges: black
cross edges: (3,2), (4,7), (5,7), (9,4)
back edge: (9,1).

root 
(wave-front propagation)

"forward edge" (u,v)?

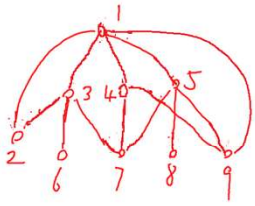

$\Rightarrow$ No, in BFS, (u,v) is a tree edge

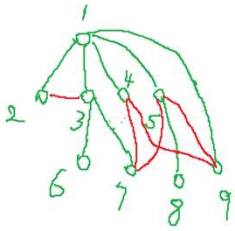
10

Similarly, we can classify edges by BFS on undirected graphs

Eg.

BFS :





tree edges:
(1,2), (1,3), (1,4), (1,5), (1,9)
(3,6), (3,7), (5,8)
cross edges:
(2,3), (4,7), (4,9), (5,7), (5,9)

"back edge" (9,1)?
⇒ No, in undirected graph
(1,9) is a tree edge

11