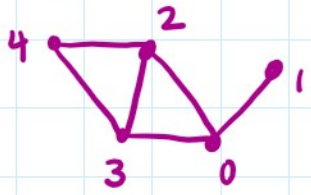
CMSC351-2023-11-01
Wednesday, November 1, 2023 8:34 AM

Graphs, Continued

① Notation: Typically:
V = # vertices (sometimes n instead)
E = # edges

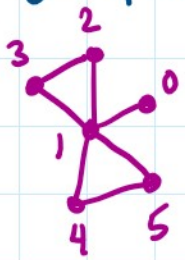
② Walks, etc

(A) defn: A **walk** is a route (intuitive sense) from a starting vertex to ending vertex (may = starting vertex) following edges
May repeat either edges or vertices.

ex 

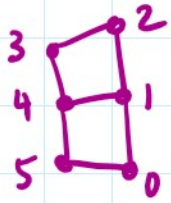
an example of a walk is:
0,1,0,3,2,3,4
listed as a sequence of vertices.

(B) defn: A **trail** is a walk w/ no repeated edges.
still may repeat vertices.

ex 

an ex. of a trail:
0,1,2,3,1,5
repeats vertex 1 but no edges.
not a trail:
0,1,2,3,1,2

(C) defn: A **path** is a walk w/ no repeated vertices
(hence we cannot have repeated edges either)

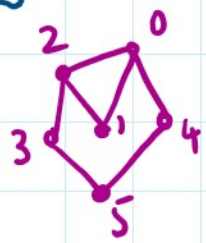
ex 

an ex. of a path:
1,4,3,2
not a path:
1,4,5,0,1,2

(D) defn: A **cycle** is a trail w/ no repeated vertices

1,4,3,0,1,2

(D) defn: A **cycle** is a trail w/ no repeated vertices except the starting vertex = ending vertex.

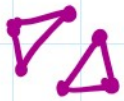
ex 

some cycles: 0,1,2,0
0,1,2,3,5,4,0

(E) defn: A graph is **connected** if, for any two vertices, ∃ path blw them.

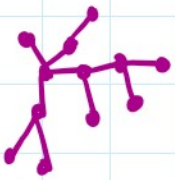
ex 

Connected

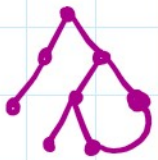
ex 

not connected

(F) defn A **tree** is a simple connected graph w/ no cycles.

ex 

tree

ex 

not a tree!

(3) Graph Storage

(A) There are several ways to store a graph. Here are two.
Assume simple, unweighted, undirected (for now).

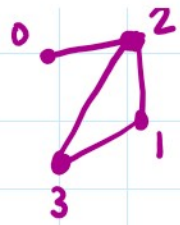
(B) Adjacency Matrix:
Sps we have V vertices labeled 0,1,...,V-1
Create a 0-indexed V×V matrix AM s.t.

$$AM[i][j] = \begin{cases} 1 & \text{if } \exists \text{ edge blw vertices } i \text{ and } j \\ 0 & \text{o/w.} \end{cases}$$

ex 

$AM = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & \cdot \end{bmatrix}$

AM[0][2]=1 bc ∃ edge blw 0,2

ex 

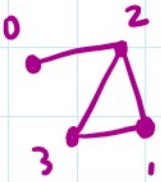
$$AM = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

note: Can be wasteful if $V >> E$ then AM has lots of 0s.

(c) Adjacency list (of lists)

Sps we have V vertices labeled $0, 1, \ldots, V-1$.
Create a list (of lists) AL s.t.

AL has V entries, each is a list.
AL[i] = list of vertices that i is adj. to.

ex 

$$AL = [[2], [2,3], [0,1,3], [1,2]]$$

note Not wasteful.
also if @ a vertex we have immediate access to adjacent vertices.

(D) Pseudocode to demonstrate

```
(For a graph G we'll do:)
for each vertex x in G:
        constant time 1
        for each vertex w adj to x:
                constant time 2
        end for
end for
```

- if we have adj matrix then outer for loop runs V times and the inner one runs V times b/c for each x we have to scan x's row in AM to look for 1s and that's V steps. $\Theta(V^2)$.

and that's V steps. $\Theta(V^2)$.

- if we have an adj list, the outer loop still V times.
  But the inner loop is not V times b/c it dep. on x.
  sneaky:
  - ignore inner loop we take $\Theta(V)$ time (constant time 1)
  - for each edge a — b the inner loop runs twice,
    once when x=a and once when x=b.
    so the inner loop takes time $\Theta(2E)$ over the
    entire run!
    Thus the time is $\Theta(V+2E)$.