1 # CS3134 #22

11/20/03

Janak J Parekh

2 # Administrivia

- Minor typo on HW#5 (points)
- Scheduling final exam?

3 # Agenda

- Minimum spanning tree
- Directed graph algorithms

4 # Complexity of BFS and DFS?

- Optimally, O(V+E) – we visit every vertex a constant number of times and potentially travel every edge a constant number of times
- But this is only for an adjacency list; in an adjacency matrix version, it's $O(V^2)$ – we scan every row and every column in the adjacency matrix once
- Admittedly inefficient, but we knew that

5 # Minimum spanning trees

- A (minimum) spanning tree is a subgraph with no cycles
  - Different in weighted graphs
- Remove graph redundancy
- Useful for many applications
  - Ex: minimize wiring
- In a minimum spanning tree, #E = #V – 1
- Simple algorithm (p. 644): DFS and record the edges traveled
  - Don't worry about backtracking
  - Can also use BFS…

6 # Directed graphs

- As earlier mentioned, useful for situations where we need to model "one-way" information
  - Streets
  - Trees are a subclass of directed graphs
  - Book: course prerequisites
- Topological sorting: come up with a legitimate ordering of processing the nodes
  - Often useful for *partial ordering* problems, such as aforementioned course prerequisites
  - Result: a order where no vertex y comes before a vertex x where x→y
  - There can be multiple correct answers!

7 # Topological sort

- Find a vertex that has no successors, i.e., arrows that point to *it*
  - Look at columns of the adjacency matrix
- Delete that vertex and print it out
- Repeat
- What kinds of graphs doesn't this work for?
  - Cycles – what happens?
  - "Catch-22" in real life

- In other words, works on generalized trees (multiple roots, etc.) – *DAG*

- Complexity again O(V+E)/O(V^2)

8 Topological sort (II)

- How to find node with no successors?
- How do you delete a node?

9 Next time

- Warshall's Algorithm
- Start weighted graphs