CS W3134: Data Structures in Java

Lecture #24: Graphs IV
12/7/04
Janak J Parekh

Administrivia
- HW#6 due on Monday
  - Any questions?
- Fill out recommendations
- Final exam review time?
  - Maybe next class

Agenda
- Graphs cont’d.
Prim revisited

- Book’s code only inserts one edge to an unvisited node given existing sources
- Then occasionally has to “update” it with a cheaper edge
- You can actually do it either way
  - If you insert all edges, when you’re ready to remove, just keep on removing until you find one to an unvisited vertex
  - By the way, I don’t like how the book describes this algorithm that much

Shortest-path problem

- Given a graph with weighted edges, and a starting vertex, find shortest path to a target
- Dijkstra’s algorithm most canonical way of doing it
- So turns out you get shortest paths to all remote vertices from that starting vertex
- Can handle both directed and undirected graphs
  - Produces a directed tree
  - Cannot handle negative weights

Dijkstra’s Algorithm: Basic idea

- Initialize an array of distances from starting node to each vertex – if there doesn’t exist a direct edge to a vertex, consider it at “infinite” distance
- Add the closest node not already in the shortest-path tree
- Update weights based on edges from newest node plus distance from starting to new – and keep track of the node we used to get to that target
- Repeat
- To find a path to a node, go backwards through the parent nodes
Floyd’s Algorithm

- For all-pairs shortest path, in $V^3$ time
- Idea based on Warshall’s algorithm, but add weights together
- For all $j_i$
  - For all columns $x$ in row $j_i$
    - If any value $(x, y)$ is 1,
    - For all rows $z$ in column $j_i$, then update $(x, z) + (x, y)$
    - Optionally, store path $(x, y)$ through $y$
- Remember, array references are “backwards”

Putting it all together...

- What have we studied?
  - Low-level structures
    - Arrays, references
  - High-level structures
    - Lists, hash tables, trees, graphs
  - Algorithms
    - Recursion
    - Insertion sort, Quick sort, Merge sort, Heap sort
  - Multiple ways to slice-and-dice
  - Book: “general-purpose” vs. “specialized”
  - Nifty tables on pgs 722, 724, 725

Intractable problems

- There are graph (and other!) problems that can’t be done in any reasonable time (linear, logarithmic, polynomial) – they’re often exponential time, e.g., $x^n$ – and grow way too quickly
- Considered NP-complete (Non-deterministic Polynomial)
- Insta-Ph.D.: prove $P=\mathbb{NP}$ (or vice-versa)
- Example: traveling salesman problem – visit all cities exactly once, and return to starting point, taking minimum-cost path
  - Hamiltonian cycle problem
  - $N!$ time!
Java data structures

- Collections (container) API
- Collections and maps
  - Collections: Sets, SortedSets and Lists
  - Maps: Map and SortedMap
- Implementations:
  - Sets: HashSet, TreeSet
  - Lists: ArrayList, LinkedList
  - Maps: HashMap, TreeMap
- Lots of utility methods
  - Sort, shuffle, search, findMax/findMin
  - Works with generic "Object"'s
  - In the real world, get comfortable with these – they work well!

The Exam

- Similar to midterm, but about 50-75% longer
- What you don't need to know
  - Shell sort
  - Red-black trees
  - 2-3-4 trees/external storage
  - Floyd's algorithm (too hard to do on the exam)
- What you do need to know
  - Pretty much everything else
  - Remember, stuff in class – use my slides
  - Chapter 15 is a useful overview

Next time

- If you see this slide on Tuesday, it means we’re done.
- Review session on Thursday?