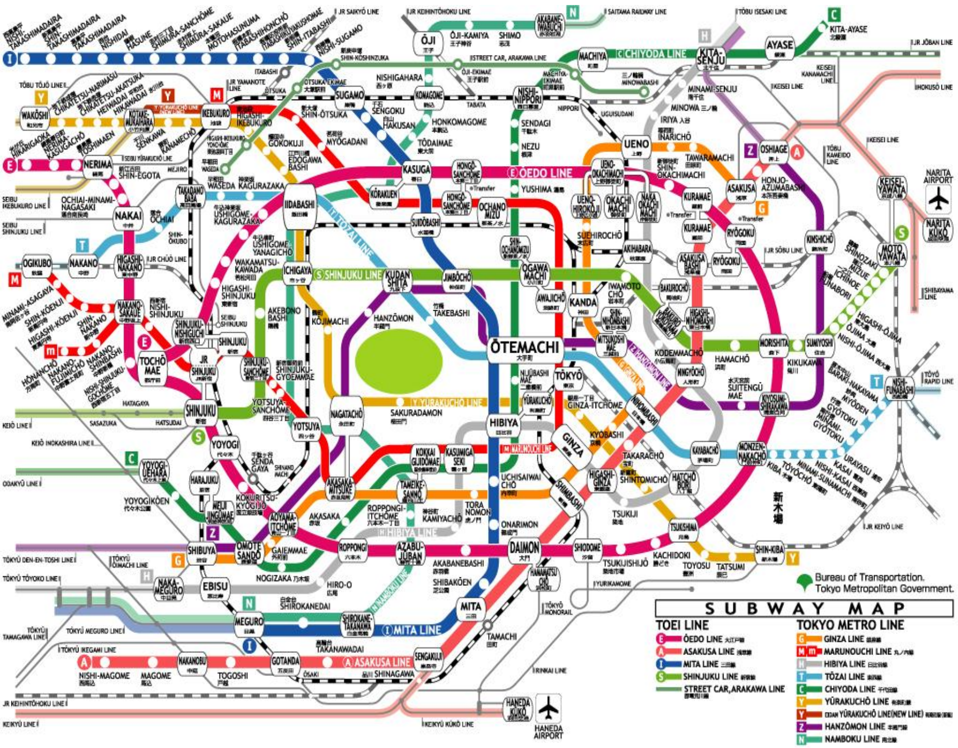
CS/ENGRD 2110
Object-Oriented Programming and Data Structures

Spring 2012

Thorsten Joachims



Lecture 19: Shortest Paths

Shortest Paths in Graphs

- Finding the shortest (min-cost) path in a graph is a problem that occurs often
  - Best flight from Ithaca, NY to Duesseldorf, Germany?
  - How closely are two people connected on Facebook?
  - Driving directions from Ithaca, NY to Queens, NY?
  - Result depends on our notion of cost
    - Number of hops
    - Least mileage
    - Least time
    - Cheapest
    - Least boring
  - All of these “costs” can be represented as edge weights
- How do we find a shortest path?

Breadth-First Search for Shortest Paths Unweighted Graphs

- Input: start node s, destination node t
- Put start s node into queue and mark s as visited.
- While queue not empty
  - Poll n off queue.
  - FOR all (unmarked) successors n’ of n
    - IF n’ equals t THEN return path
    - Put n’ into queue
    - Mark n’ as visited.
- Time complexity:
  - O(m) time

Why does BFS find Shortest Path?

- Any node in distance 1 is visited before any node at 2 hops, before any node at distance 3 hops, …
- Whenever a node is at the top of the queue for the first time, we must have gotten there with the minimum number of hops.
- How do we keep track of the path that got BFS there?
  - Store predecessor node on path for each node in graph.

Breadth-First Search for Shortest Paths Weighted Graphs

- Input: start node s, destination node t
- Put start (s,0,null) into min-priority queue.
- Initialize empty dictionary **`path`**.
- While queue not empty
  - Poll minimum element (n,c,prev) off queue.
  - Mark n as "done" in **`path`** by storing prev.
  - IF n equals t THEN return **`path`**
  - IF n is not yet "done"
    - FOR all successors n' of n that are not "done"
      - Put (n',c+weight(n,n'),n) into priority queue
- Time complexity:
  - O(m log m) time using heap and adjacency lists
  - Can be improved