Administrative Details

- Final exam - self scheduled
- You get 2.5 hours to complete it
- Covers everything, with strong emphasis on Ch 14-16 (BSTs, HashTables, Maps, Graphs)
- Study guide on handouts page
- Makeup class today 1:10-2 in Wege
- Lab 11 is also today (optional) from 2-4
- You’ll get midterms and Lab 9 (Darwin) back this afternoon (I’ll grade Lab 10 next week)
- Extra credit accepted through Tue, May 20 at 5pm
- I will be out of town this weekend

Last Time

- Briefly discussed DFS and BFS
- Darwin
  - Nice job everyone
  - Congrats to Riley!

Today’s Outline

- Finish discussing graph traversal algorithms
  - Cycle detection
  - Dijkstra’s algorithm (least cost shortest path)

Shortest Path and Cycle Detection

- Shortest path
  - How do we find shortest path from src to all nodes?
    - Could find all paths and then pick smallest…
    - …this is bad—there are many paths (O(n!))!
  - Can we use BFS or DFS to find shortest path to all nodes?
    - BFS (with labels on nodes that indicate “cost” from src)
  - Cycle detection
    - Is there a path starting at src that contains a cycle?
    - Should we use BFS or DFS?
      - DFS (stack tells us path from src to current node; see if path leads to node already on stack)
Shortest Path (No Weights on Edges)

TCL

West

Bronfman

Jesup

Art

Griffin

School

Pool

TBL

0

todo queue

Add 1 to neighbors of TCL...

Shortest Path (No Weights on Edges)

Jesup

Bronf

West

Art

Griffin

School

Pool

TBL

1

0

todo queue

Add 1 to neighbors of Bronf...

Shortest Path (No Weights on Edges)

Jesup

West

Bronf

Art

Griffin

School

Pool

TBL

2

2

2

2

todo queue

Add 1 to neighbors of Jesup...

Shortest Path (No Weights on Edges)

Art

West

Bronf

Jesup

Art

Griffin

School

Pool

TBL

1

1

2

2

todo queue

Add 1 to neighbors of Jesup...
Shortest Path
(No Weights on Edges)

Add 1 to neighbors of Art...

Cycle Detection

Partially completed DFS...
Shortest Path Revisited

- What if there are weights on our edges?
- Will BFS still work?
  - No! BFS processes nodes according to number of edges/hops from src to node
- Need something else…
Dijkstra's Algorithm

- Basic idea
  - Explore paths from src in order of increasing total cost

- Algorithm
  - Keep map from node to shortest-distance-to-node
  - Keep PQ of paths from SRC, ordered by total distance
  - While PQ is not empty
    - Take shortest path SRC -> ... -> DEST from PQ
      - If DEST has not been visited
        - Fix distance to DEST in map
        - Extend path to all neighbors of DEST
        - Add new paths to PQ

Priority Queue
More Graph Algorithms!

- Topological sorting (DFS)
  - Only valid for directed and acyclic graphs
  - List vertices in such a way as to make the edges point in one direction (i.e., back to beginning)
  - Commonly used to solve scheduling problems, assembly lines steps, etc.
- Minimum cost spanning tree (MST)
  - Find edges with least total weight that connects all nodes
  - Not quite DFS or BFS…uses a greedy algorithm to select edges
  - Important for phone, cable, water systems, etc.

Topological Sort: Unicycle Factory

```
Attach Seat to Frame
Paint Frame
Attach Wheel to Frame
Attach Pedals to Wheel
Inflate Tire

Finished Unicycle
```

```
Attach Seat
Paint Frame
Attach Wheel
Inflate Tire

Finished Unicycle
```

```
Inflate Tire
Attach Pedals
Paint Frame
Weld Frame

Finished Unicycle
```

```
Inflate Tire
Attach Pedals
Paint Frame
Weld Frame

Finished Unicycle
```

```
Attach Wheel
Attach Seat
Finished Unicycle
Inflate Tire

Finished Unicycle
```

```
Attach Wheel
Attach Seat
Finished Unicycle
Inflate Tire

Finished Unicycle
```

```
Attach Wheel
Attach Pedals
Paint Frame
Weld Frame

Finished Unicycle
```

```
Attach Wheel
Attach Pedals
Paint Frame
Weld Frame

Finished Unicycle
```

```
Attach Wheel
Attach Pedals
Paint Frame
Weld Frame

Finished Unicycle
```

```
Attach Wheel
Attach Pedals
Paint Frame
Weld Frame

Finished Unicycle
```

Minimum Spanning Tree (MST)

Moving on...Maps!
- Maps are data structures that map keys to values
  - Sometimes called dictionaries
  - Where have we seen maps before?
    - Associations!
    - An Association is a single key-value pair. Maps may contain lots of key-value pairs. (Kinds like a Vector of Associations, but better!)
- In Java, Maps are found in java.util package
- What methods are needed in the Map interface?

Map Interface
- Methods for Map<K, V>:
  - int size() - returns number of entries in map
  - boolean isEmpty() - true iff there are no entries
  - boolean containsKey(K key) - true iff key exists in map
  - boolean containsValue(V val) - true iff val exists at least once in map
  - V get(K key) - get value associated with key
  - V put(K key, V val) - insert mapping from key to val, returns value replaced (old value) or null
  - V remove(K key) - remove mapping from key to val
  - void clear() - remove all entries from map

Map Interface
- Other methods for Map<K, V>:
  - void putAll(Map<K, V> other) - puts all key-value pairs from Map other in map
  - Set<K> keySet() - return set of keys in map
  - Set<Association<K, V>> entrySet() - return set of key-value pairs from map
  - Structure<V> valueSet() - return set of values
  - boolean equals() - used to compare two maps
  - int hashCode() - returns hash code associated with map (stay tuned...)

Map Interface
Sample Usage

• See Dictionary.java

Simple Map Implementation

• A simple implementation of the Map interface is the MapList class
• Uses a SinglyLinkedList of Associations as underlying data structure
• How would we implement put(K key, V val)?
• What is the running time of:
  • containsKey(K key)?
  • containsValue(V val)?
• Bottom line: not O(1)!

Search/Locate Revisited

• How long does it take to search for objects in Vectors and Lists?
  • O(n) on average
• How about in BSTs?
  • O(log n)
• Can this be improved?
  • With hash tables, YES!
  • Can locate objects in roughly O(1) time
• …to be continued!