3134 Data Structures in Java

Lecture 20
Apr 16 2007
Shlomo Hershkop
Announcements

- Homeworks posted
- Theory due in 2 weeks
- EC theory due in 2.5
- Programming in 3ish
Today

- Disjointed set DS
- MST
- Network flows
DS : Disjointed Sets

- Many times want to know if things are related to each other in some way

- Relationships might change over time

- Many graph algorithms are based on these ideas
  - MST
Disjointed sets

- Given a bunch of elements I want to be able to organize them into sets (collections) and be able to add and find specific elements
- An element can only be part of one set at a time

- Union
- Find
Equivalence Class ADT

- Union(a,b)
  - merges 2 equivalence classes

- Find(a)
  - retrieves equivalence class containing a
Implementation?

- Any ideas on implementing this DS?

- How long to do UNION??

- How long to do FIND??
Arrays

- use an array and store name of class at each position

- example of union

- what is the running time
  - find?
  - union?
running times

- so find we can do in $O(1)$

- merge, worst could be $O(N)$
  - for $M$ merges on $N$ items
    - $O(NM)$
    - $M$ can’t exceed $N$
    - so $O(N^2)$
so there must be a better way

something to notice:
1. the name of the elements are static
2. the name of the set is arbitrary
solution

- use tree type relationship
  - here it is graphically

- forest:
  - collection of trees
- element which sits at the top is the root
- root will be the name of the class
- find spits out the root name as the set name
- union(a,b) makes b a child of a
implementation

- forest can be an array with each item containing the index of the parent
- -1 could be no parent, hence root
- constructor puts -1 in each element’s spot

- lets do some code
find

int find(int N) {
    if(set[N] < 0)
        return N
    else
        return find(set[N])
}
void union(int item1, int item2) {
    int root1, root2;
    root1 = find(item1);
    root2 = find(item2);
    if(root1 != root2)
        set[root2] = root1;
}
- runtimes ??
- `find(n)`
- `union(1)`

- haven’t really improved it much
tricks

- so need a trick to speed things up by not having deep trees
  - Union by size
  - Union by height
Union by size

- Rule: always make the SMALLER tree a subtree of larger one

- Will need to keep track of how many nodes in set
  - Why is that??
  - ideas ??
Union by height

- Rule: always make shallower tree a subtree of the deeper one

- how will the height change ??

- can you code this ?
Back to graphs
graph issues

- make sure you are clear on:
  - how would you represent a graph from the code point of view
  - how would you save a graph?
  - load a graph?
- Code example on board
- City java object
- Disjointed sets class
- Please try to code by next class