CSE 326: Data Structures
Disjoint Set Union/Find

Brian Curless

Spring 2008

Announcements (5/19/08)

- Homework due on Friday at the beginning of class.
- Reading for this lecture: Chapter 8.

Making Connections

You have a set of nodes (numbered 1-9) on a network. You are given a sequence of pairwise connections between them:

3-7
8-2
1-6
5-7
4-8
3-5

Q: Are nodes 2 and 4 (indirectly) connected?
Q: How about nodes 3 and 8?
Q: Are any of the paired connections redundant due to indirect connections?
Q: How many sub-networks do you have?

Making Connections

Answering these questions is much easier if we create disjoint sets of nodes that are connected:

Start: ~~{1}~~ ~~{2}~~ ~~{3}~~ ~~{4}~~ ~~{5}~~ ~~{6}~~ ~~{7}~~ ~~{8}~~ {9}
3-5 ~~{3,5}~~
4-2 ~~{2,4}~~
1-6 {1,6}
5-7 {3,5,7}
4-8 {2,4,8}
3-5

Q: Are nodes 2 and 4 (indirectly) connected?
Q: How about nodes 3 and 8?
Q: Are any of the paired connections redundant due to indirect connections?
Q: How many sub-networks do you have?

Applications of Disjoint Sets

Maintaining disjoint sets in this manner arises in a number of areas, including:

- Networks
- Transistor interconnects
- Compilers
- Image segmentation
- Building mazes (this lecture)
- Graph problems
  - Minimum Spanning Trees (upcoming topic in this class)

Disjoint Set ADT

- Data: set of pairwise **disjoint sets**.
- Required operations
  - **Union** – merge two sets to create their union
  - **Find** – determine which set a given item appears in
- A common operation sequence:
  - Connect two elements if not already connected:

    if (Find(x) != Find(y)) then Union(~~x~~,~~y~~)

    Find(x), Find(y)

Disjoint Sets and Naming

- Maintain a set of pairwise disjoint sets.
  - {3,5,7} , {4,2,8}, {9}, {1,6}
- Each set has a unique name: one of its members (for convenience)
  - {3,5,7} , {4,2,8}, {9}, {1,6}

Union

- Union(x,y) – take the union of two sets named x and y
  - {3,5,7} , {4,2,8}, {9}, {1,6}
  - Union(5,1)

    {3,5,7,1,6}, {4,2,8}, {9},

Find

- Find(x) – return the name of the set containing x.
  - {3,5,7,1,6}, {4,2,8}, {9},
  - Find(1) = 5
  - Find(4) = 8

Example

S
{1,2,7,8,9,13,19}
{3}
{4}
{5}
{6}
{10}
{11,17}
{12}
{14,20,26,27}
{15,16,21}
.
.
{22,23,24,29,39,32
33,34,35,36}

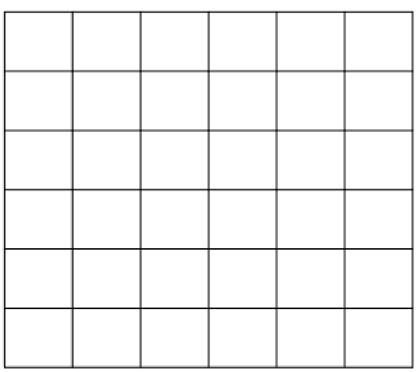
Find(8) = 7
Find(14) = 20

Union(7,20)

S
{1,2,7,8,9,13,19,14,20 26,27}
{3}
{4}
{5}
{6}
{10}
{11,17}
{12}
{15,16,21}
.
.
{22,23,24,29,39,32
33,34,35,36}

Nifty Application: Building Mazes

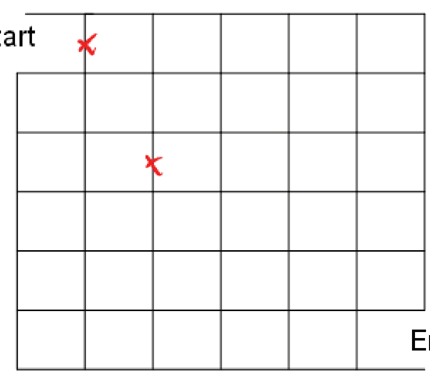
Idea: Build a random maze by erasing edges.



Building Mazes

- Pick Start and End

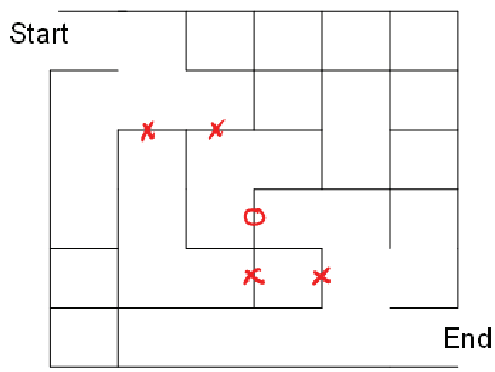
Start



End

Building Mazes

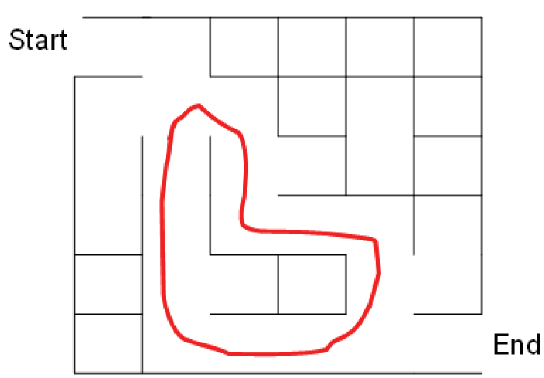
- Repeatedly pick random edges to delete.



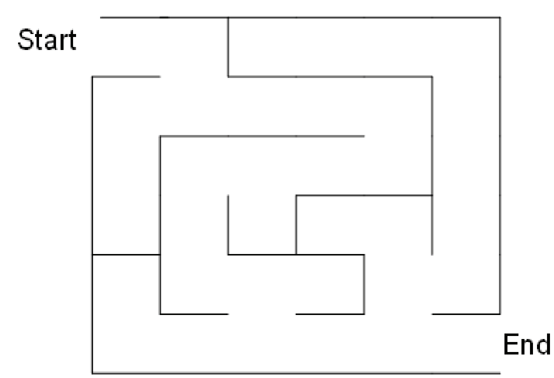
Desired Properties

- None of the boundary is deleted (except at "start" and "end").
- Every cell is reachable from every other cell.
- There are no cycles – no cell can reach itself by a path unless it retraces some part of the path.

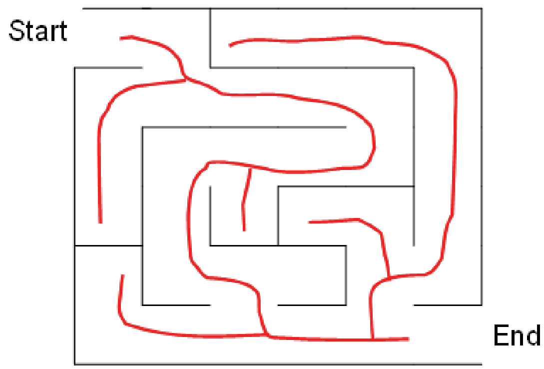
A Cycle



A Good Solution



A Hidden Tree



Number the Cells

We start with disjoint sets S ={ {1}, {2}, {3}, {4},… {36} }.
We have all possible edges between neighbors
E ={ (1,2), (1,7), (2,8), (2,3), … } 60 edges total.

Start

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

End

Idea: Union-find operations will be done on cells.

Maze Building with Disjoint Union/Find

Algorithm sketch:

1. Choose edge at random.
   → *Boundary edges are not in edge list, so left alone*
2. Erase it (and its wall) if the neighbors are in disjoint sets.
   → *Avoids cycles*
3. Take union of those sets.
4. Go to 1, iterate until there is only one set.
   → *Every cell reachable from every other cell.*

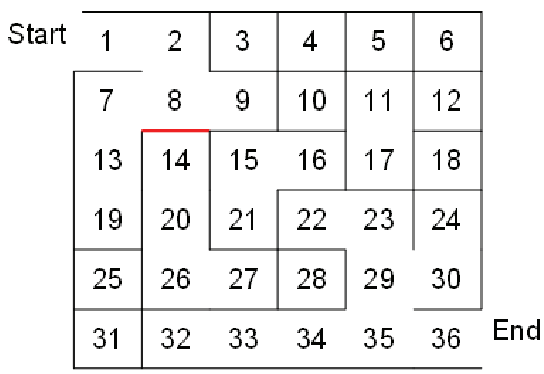
Pseudocode

- S = set of sets of connected cells
- E = set of edges
- Maze = set of maze edges initially empty

```
While there is more than one set in S
  Pick a random edge (x,y) and remove from E
  u = Find(x);
  v = Find(y);
  if u ≠ v then
    Union(u,v)
  else
    Add edge (x,y) to Maze
All remaining members of E together with Maze form
the maze
```

Example Step

Pick (8,14)



S
{1,2,7,8,9,13,19}
{3}
{4}
{5}
{6}
{10}
{11,17}
{12}
{14,20,26,27}
{15,16,21}
.
.
{22,23,24,29,30,32
33,34,35,36}

Example

S
{1,2,7,8,9,13,19}
{3}
{4}
{5}
{6}
{10}
{11,17}
{12}
{14,20,26,27}
{15,16,21}
.
.
{22,23,24,29,39,32
33,34,35,36}

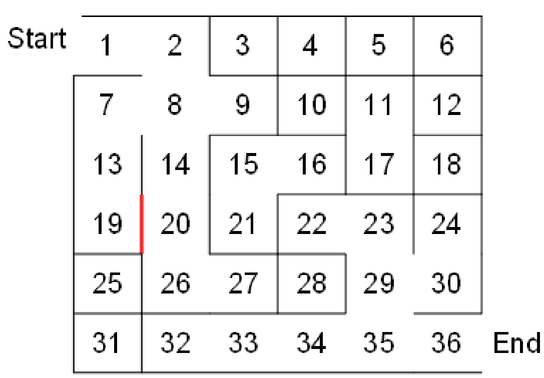
Find(8) = 7
Find(14) = 20

Union(7,20)

S
{1,2,7,8,9,13,19,14,20 26,27}
{3}
{4}
{5}
{6}
{10}
{11,17}
{12}
{15,16,21}
.
.
{22,23,24,29,39,32
33,34,35,36}

Example

Pick (19,20)



S
{1,2,7,8,9,13,19
14,20,26,27}
{3}
{4}
{5}
{6}
{10}
{11,17}
{12}
{15,16,21}
.
.
{22,23,24,29,39,32
33,34,35,36}

Example at the End

S
{1,2,3,4,5,6,7,... 36}