Application: Image Processing

One common application is in image processing

Suppose you are attempting to recognize similar features within an image

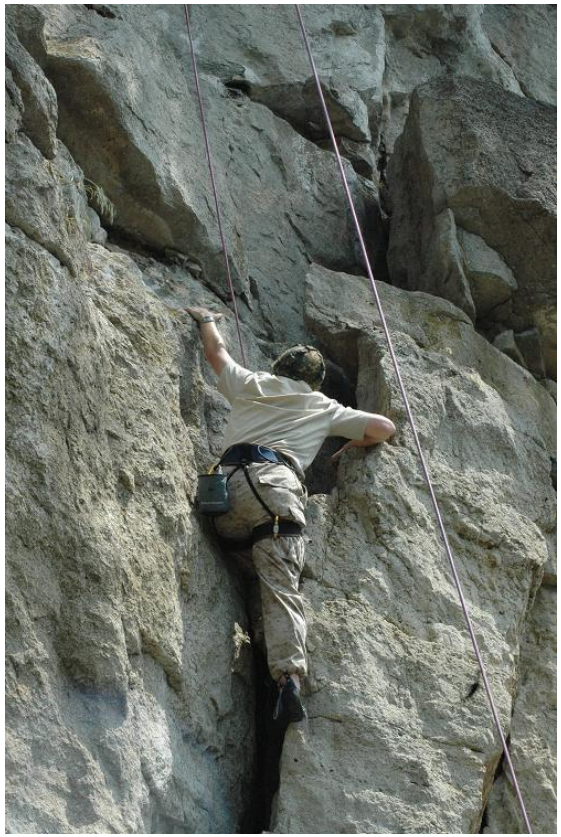
Within a photograph, the same object may be separated by an obstruction; e.g., a road may be split by

- a telephone pole in an image
- an overpass on an aerial photograph

Application: Image Processing

Consider the following image of the author climbing up the Niagara Escarpment at Rattlesnake Point

Suppose we have a program which recognizes skin tones

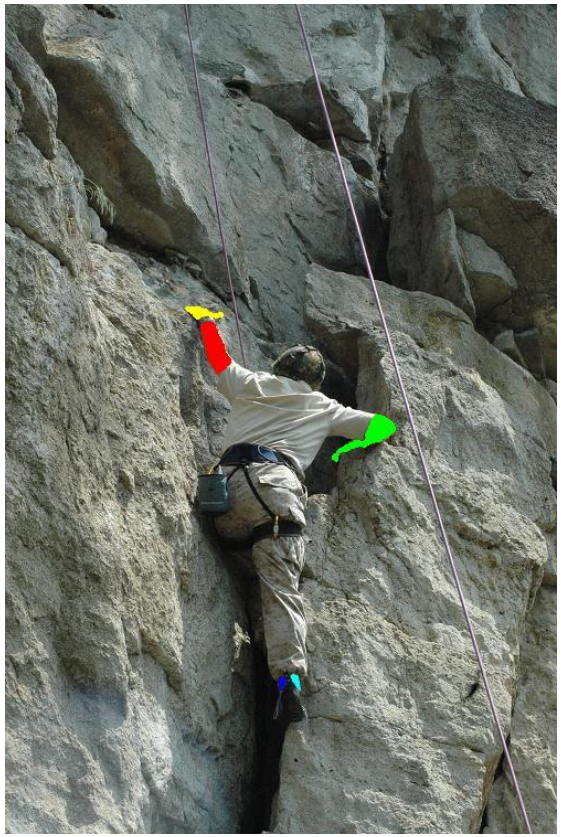


Application: Image Processing

A first algorithm may make an initial pass and recognize five different regions which are recognized as exposed skin

- the left arm and hand are separated by a watch

Each region would be represented by a separate disjoint set

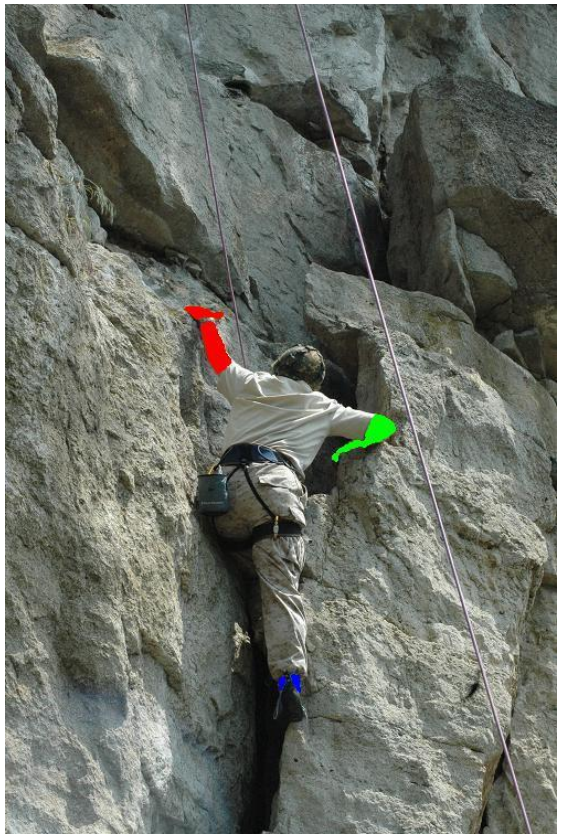


Application: Image Processing

Next, a second algorithm may take sets which are close in proximity and attempt to determine if they are from the same person

In this case, the algorithm takes the union of:

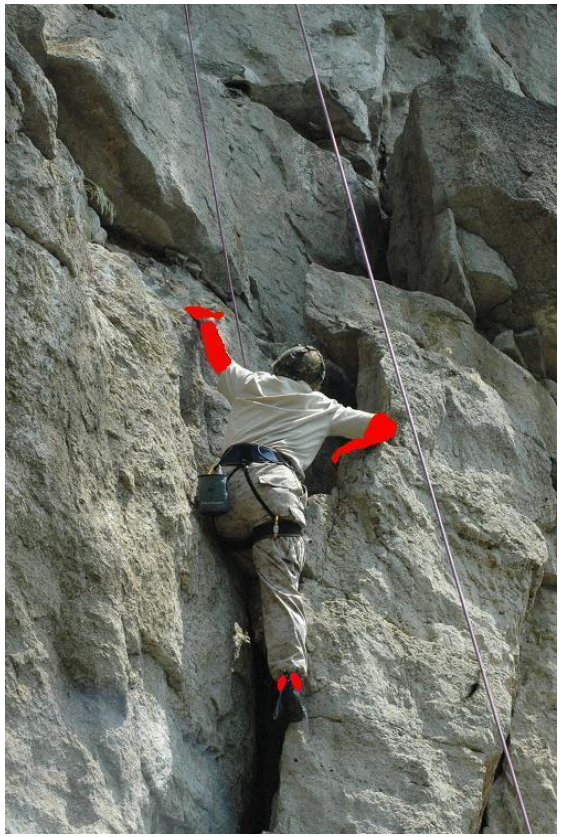
- the red and yellow regions, and
- the dark and light blue regions



Application: Image Processing

Finally, a third algorithm may take more distant sets and, depending on skin tone and other properties, may determine that they come from the same individual

In this example, the third pass may, if successful, take the union of the red, blue, and green regions



Application: Maze Generation

Another fun application is in the generation of mazes

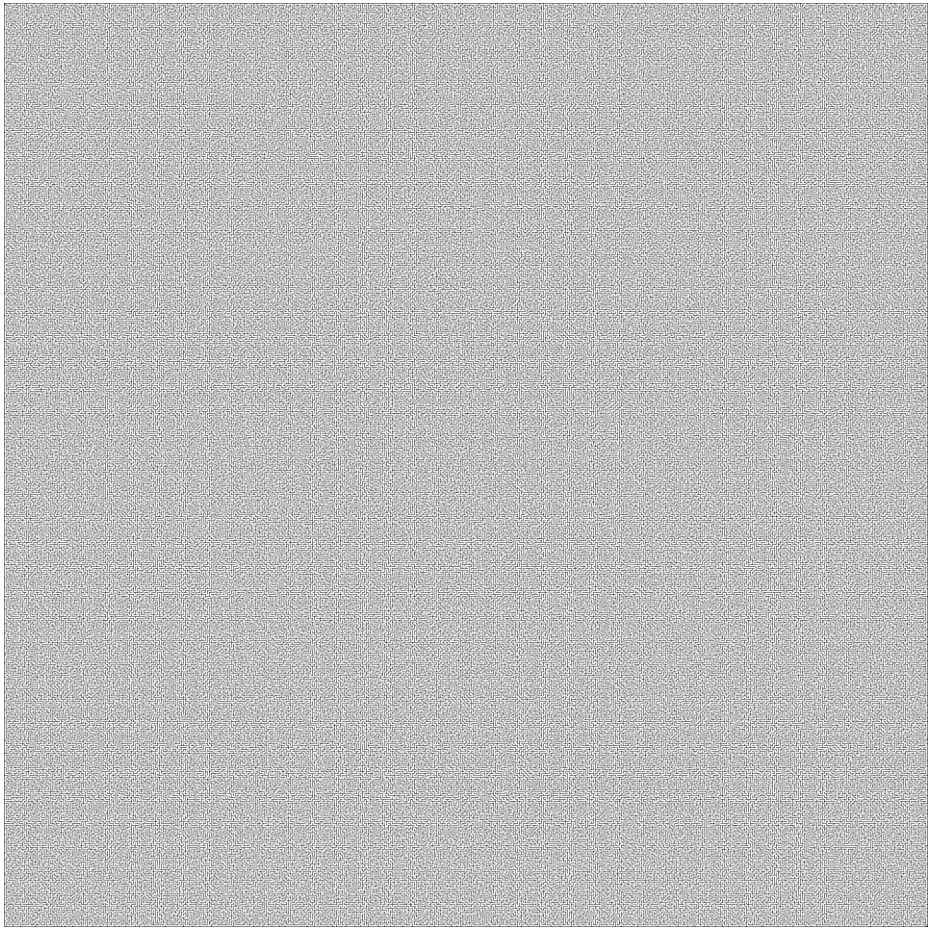
Impress your (non-engineering) friends

- They'll never guess how easy this is...

Application: Maze Generation

Here we have a maze which spans a 500 × 500 grid of squares where:

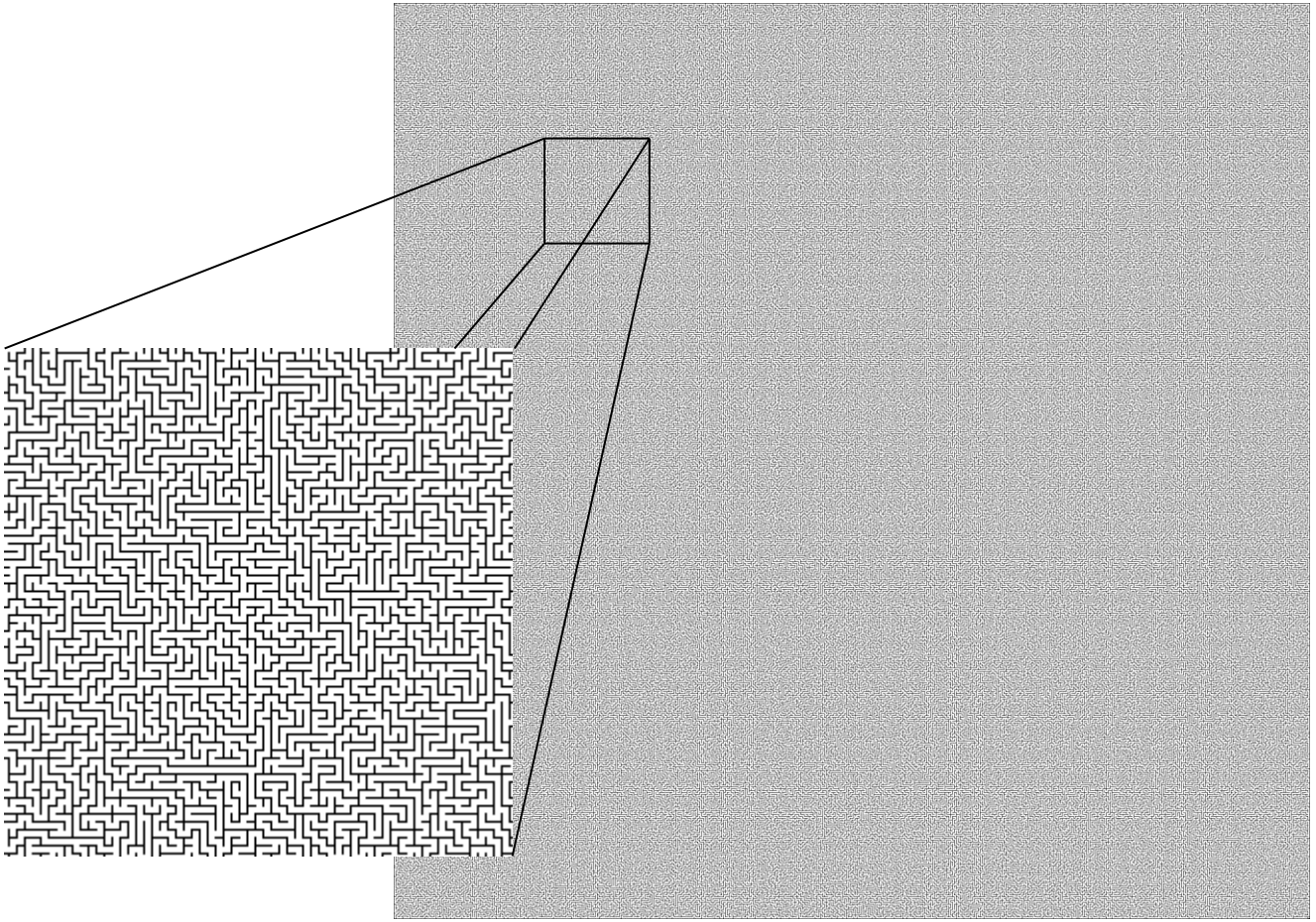
- There is one unique solution
- Each point can be reached by one unique path from the start



Ref: Lance Hampton http://littlebadwolf.com/mazes/

Application: Maze Generation

Zooming in on the maze, you will note that it is rather complex and seemingly
random



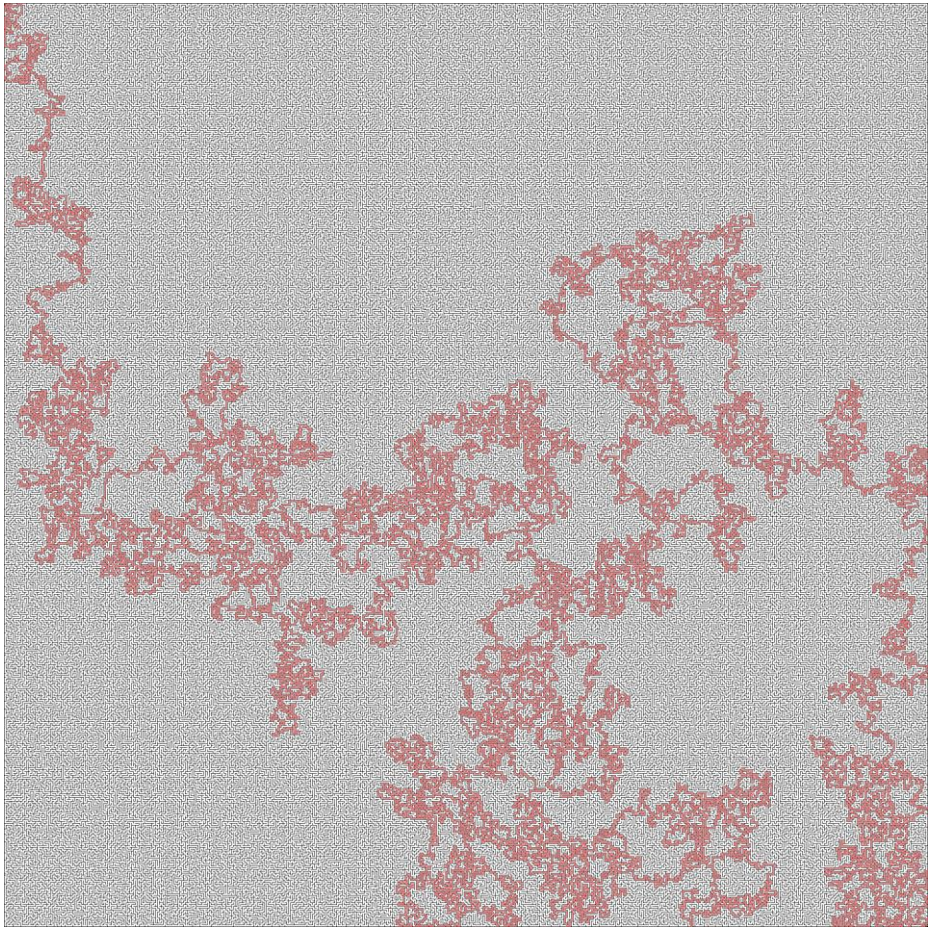
Ref: Lance Hampton http://littlebadwolf.com/mazes/

Application: Maze Generation

Finding the solution is a problem for a different lecture

- Backtracking algorithms

We will look at creating the maze using disjoint sets



Ref: Lance Hampton http://littlebadwolf.com/mazes/

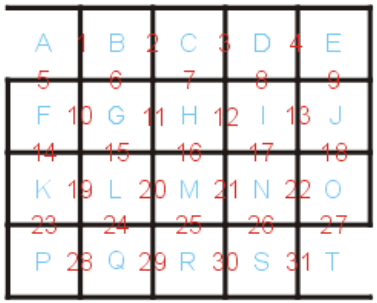
Application: Maze Generation

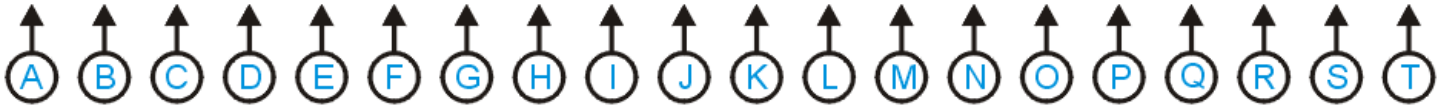
What we will do is the following:

- Start with the entire grid subdivided into squares
- Represent each square as a separate disjoint set
- Repeat the following algorithm:
  - Randomly choose a wall
  - If that wall connects two disjoint set of cells, then remove the wall and union the two sets
- To ensure that you do not randomly remove the same wall twice, we can have an array of unchecked walls

Application: Maze Generation

Let us begin with an entrance, an exit, and a disjoint set of 20 squares and 31 interior walls

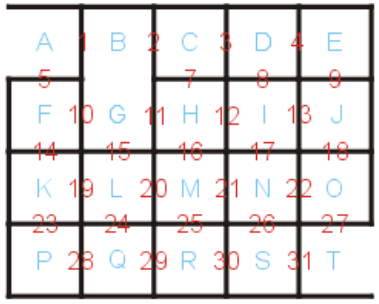


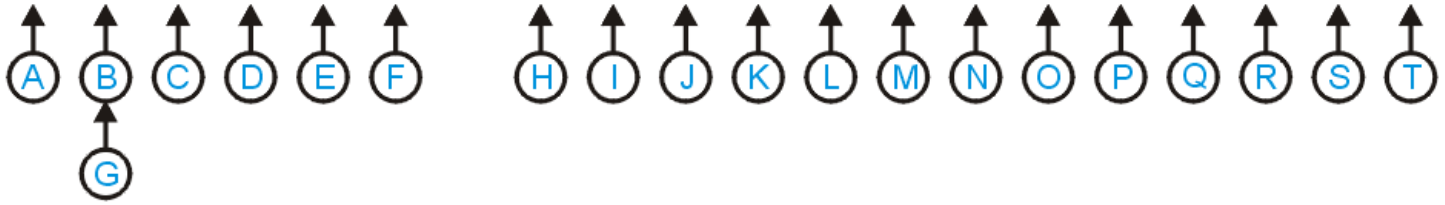


Application: Maze Generation

First, we select 6 which joins cells B and G

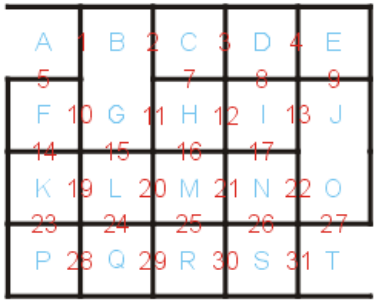
- Both have height 0

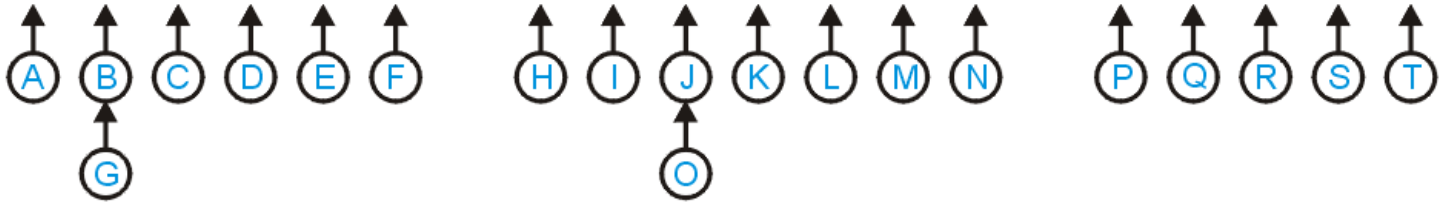




Application: Maze Generation

Next we select wall 18 which joins regions J and O



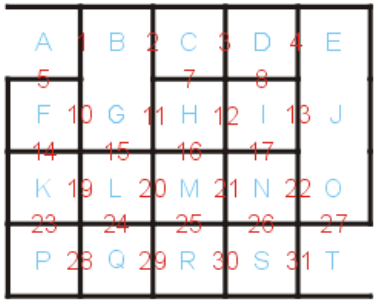


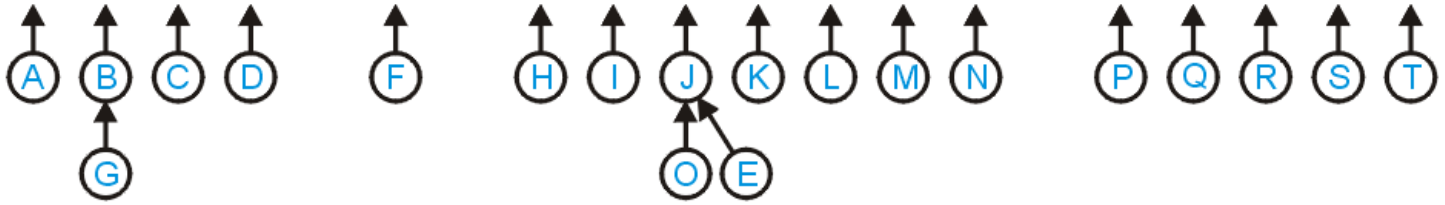
0	1	2	3	4	5	1	7	8	9	10	11	12	13	9	15	16	17	18	19

Application: Maze Generation

Next we select wall 9 which joins the disjoint sets E and J

- The disjoint set containing E has height 0, and therefore it is attached to J



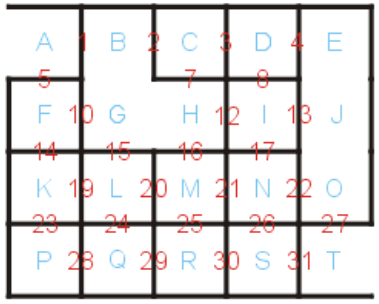


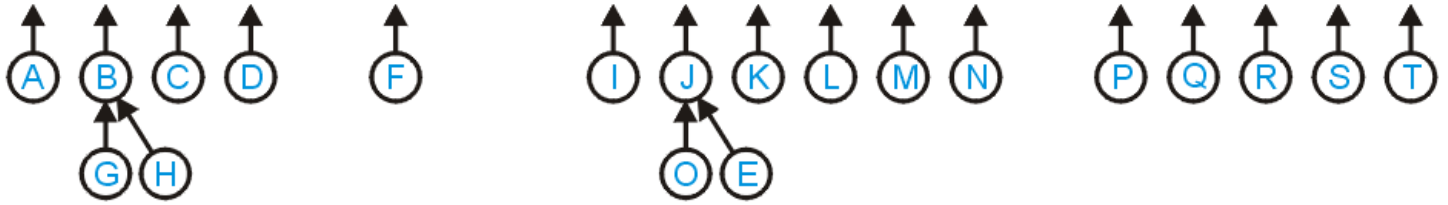
0	1	2	3	9	5	1	7	8	9	10	11	12	13	9	15	16	17	18	19

Application: Maze Generation

Next we select wall 11 which joins the sets identified by B and H

- H has height 0 and therefore we attach it to B



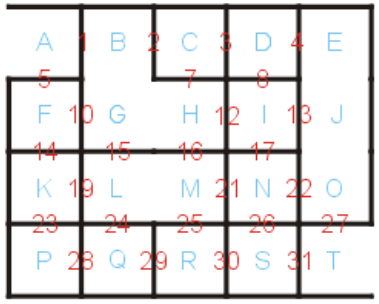


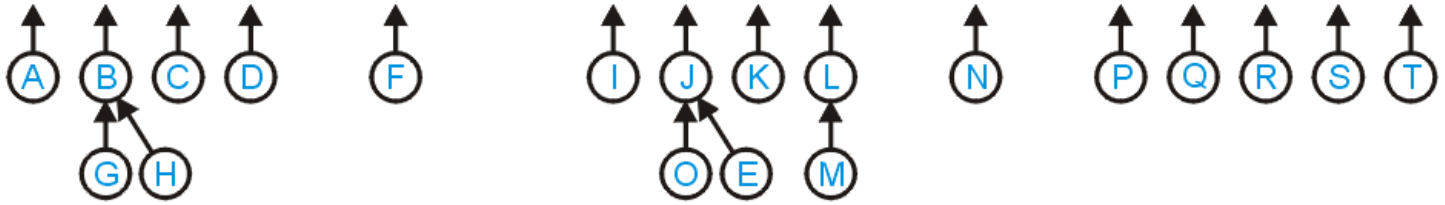
0	1	2	3	9	5	1	1	8	9	10	11	12	13	9	15	16	17	18	19

Application: Maze Generation

Next we select wall 20 which joins disjoint sets L and M

- Both are height 0



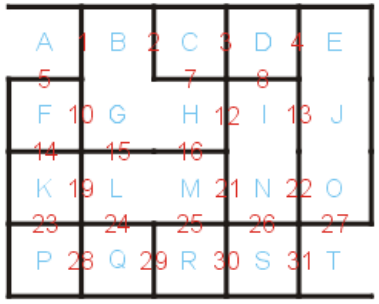


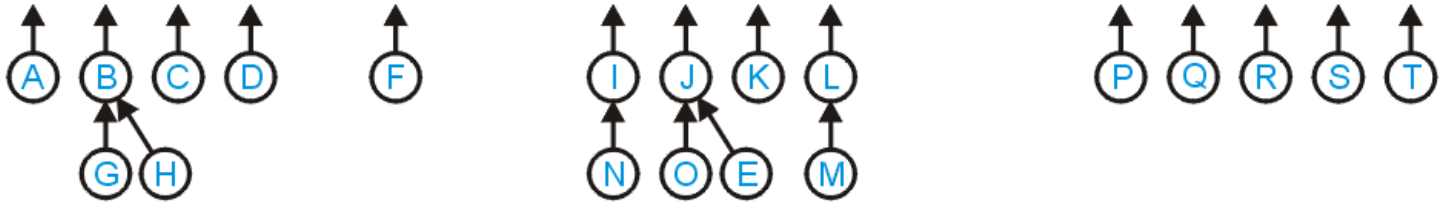
0	1	2	3	9	5	1	1	8	9	10	11	**11**	13	9	15	16	17	18	19

Application: Maze Generation

Next we select wall 17 which joins disjoint sets I and N

- Both are height 0



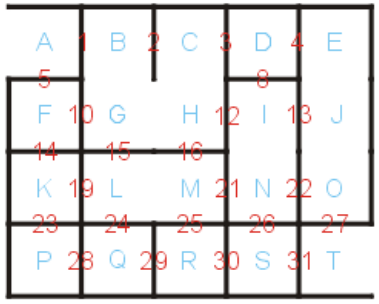


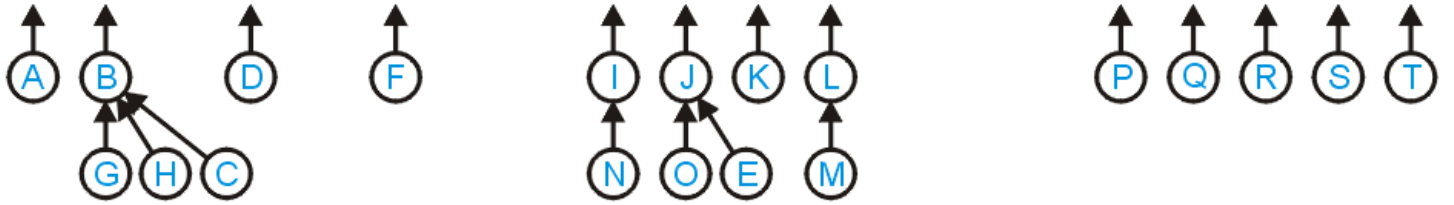
0	1	2	3	9	5	1	1	8	9	10	11	11	8	9	15	16	17	18	19

Application: Maze Generation

Next we select wall 7 which joins the disjoint set C and the disjoint set identified by B

- C has height 0 and thus we attach it to B



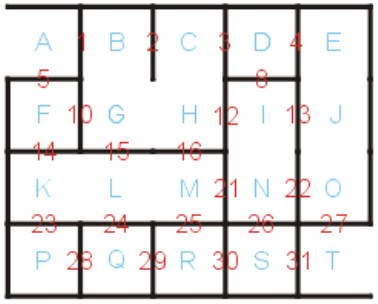


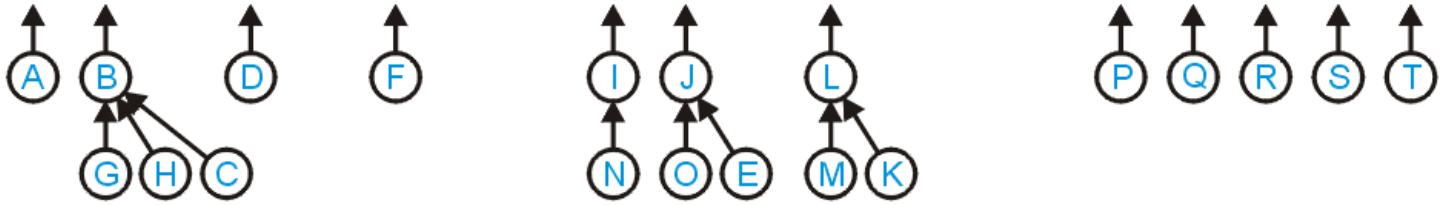
0	1	1	3	9	5	1	1	8	9	10	11	11	8	9	15	16	17	18	19

Application: Maze Generation

Next we select wall 19 which joins the disjoint set K to the disjoint sent identified by L

- Because K has height 0, we attach it to L



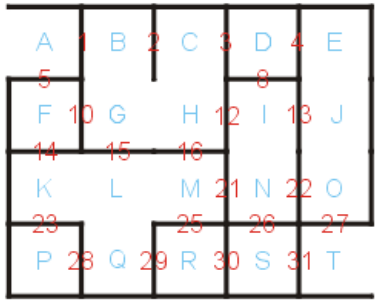


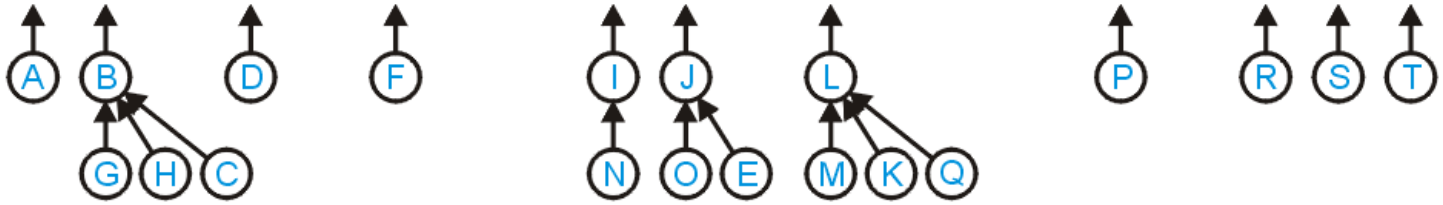
0	1	1	3	9	5	1	1	8	9	11	11	11	8	9	15	16	17	18	19

Application: Maze Generation

Next we select wall 23 and join the disjoint set Q with the set identified by L

- Again, Q has height 0 so we attach it to L



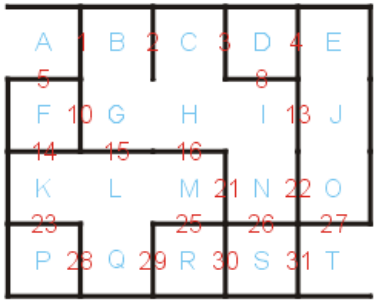


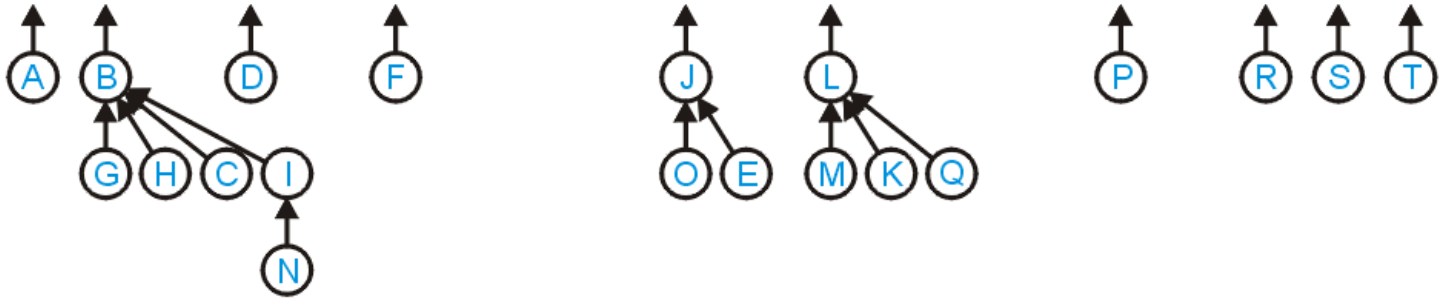
0	1	1	3	9	5	1	1	8	9	11	11	11	8	9	15	11	17	18	19

Application: Maze Generation

Next we select wall 12 which joints the disjoint sets identified by B and I

- They both have the same height, but B has more nodes, so we add I to the node B



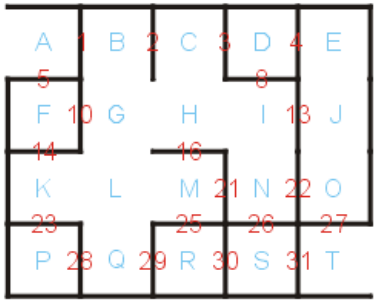


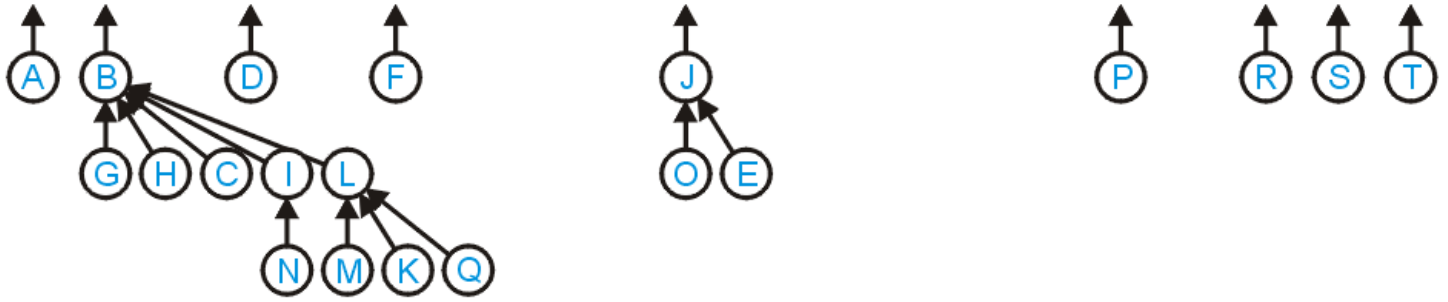
0	1	1	3	9	5	1	1	1	9	11	11	11	8	9	15	11	17	18	19

Application: Maze Generation

Selecting wall 15 joints the sets identified by B and L

- The tree B has height 2 while L has height 1 and therefore we attach L to B



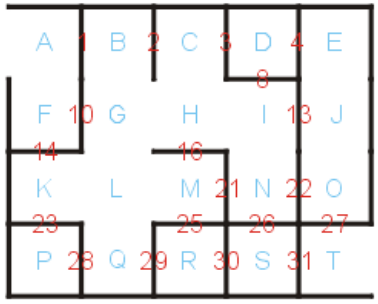


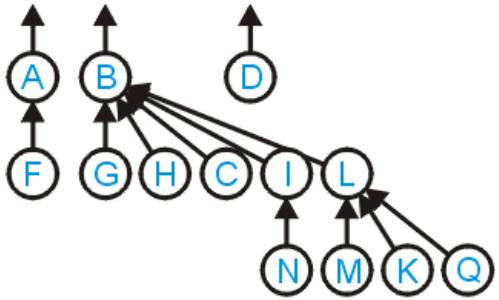
0	1	1	3	9	5	1	1	1	9	11	1	11	8	9	15	11	17	18	19

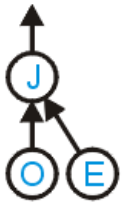
Application: Maze Generation

Next we select wall 5 which joins disjoint sets A and F

- Both are height 0



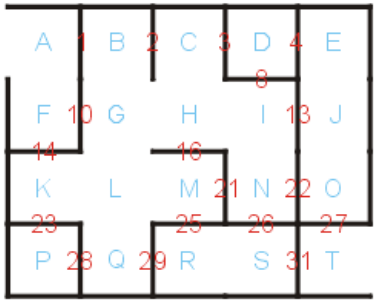


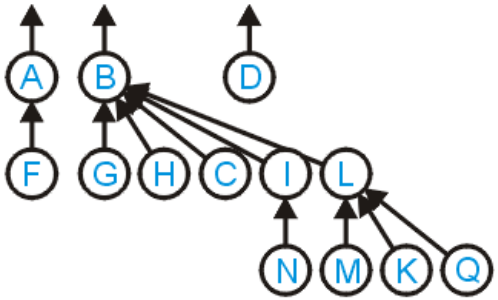


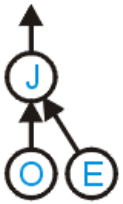
0	1	1	3	9	0	1	1	1	9	11	1	11	8	9	15	11	17	18	19

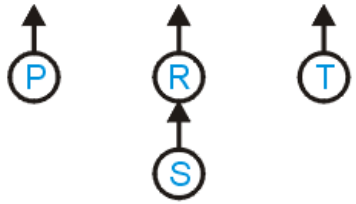
Application: Maze Generation

Selecting wall 30 also joins two disjoint sets R and S







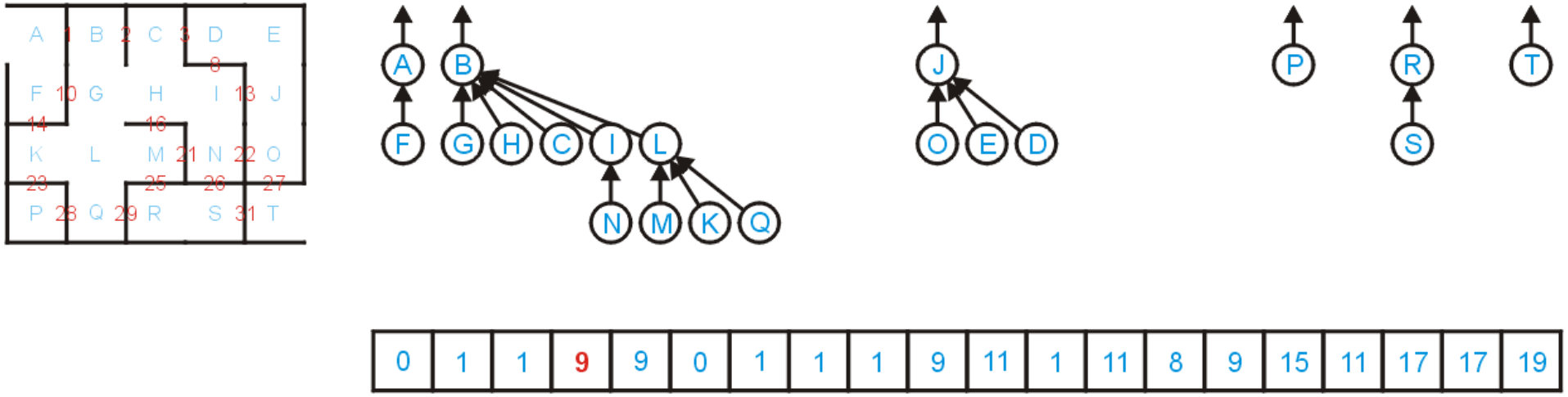


0	1	1	3	9	0	1	1	1	9	11	1	11	8	9	15	11	17	17	19

Application: Maze Generation

Selecting wall 4 joints the disjoint set D and the disjoint set identified by J

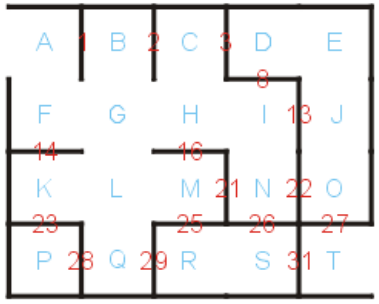
- D has height 0, J has height 1, and thus we add D to J

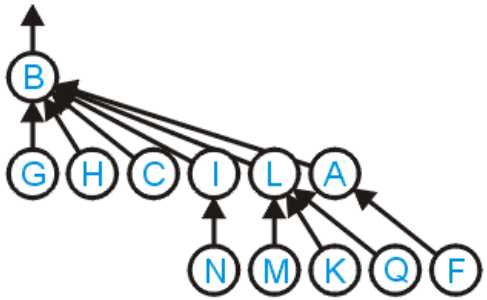


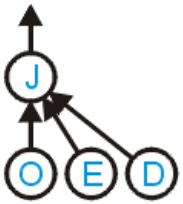
Application: Maze Generation

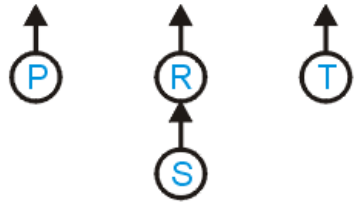
Next we select wall 10 which joins the sets identified by A and B

- A has height 1 while B has height 2, so we attach A to B







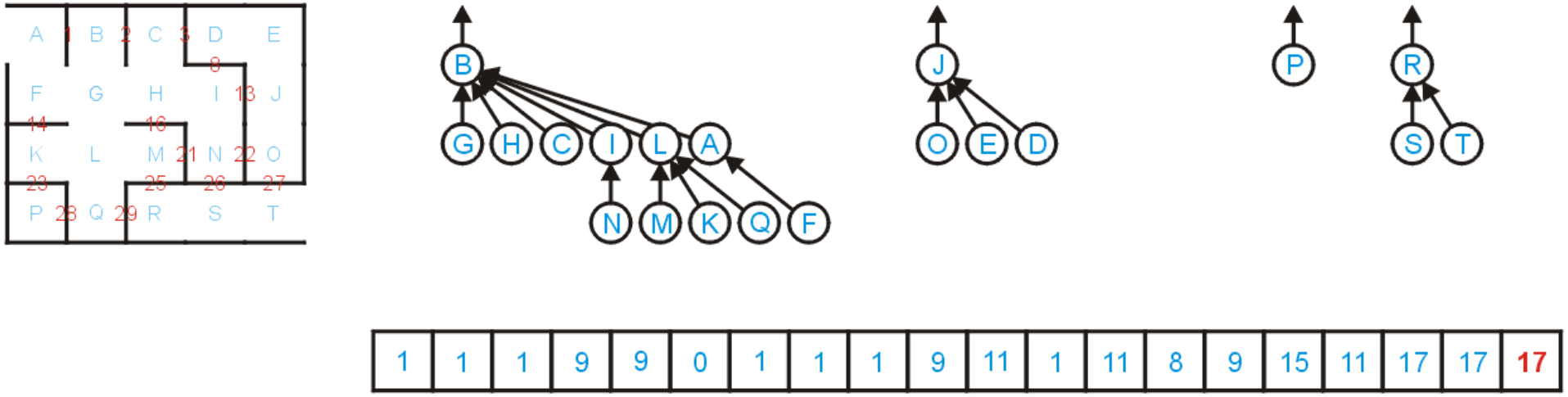


1	1	1	9	9	0	1	1	1	9	11	1	11	8	9	15	11	17	17	19

Application: Maze Generation

Selecting wall 31, we union the sets identified by R and T

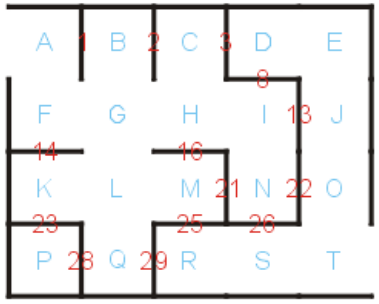
- T has height 0 so we attach it to I

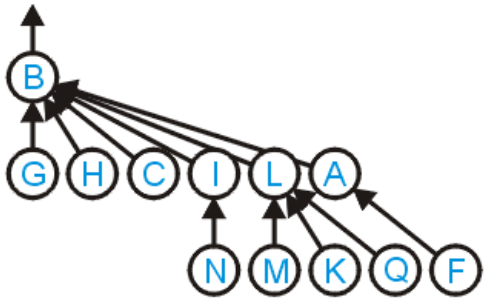


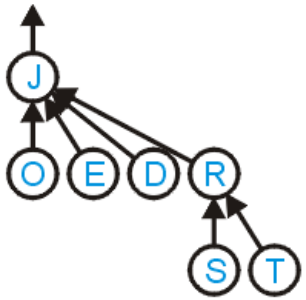
Application: Maze Generation

Selecting wall 27 joins the disjoint sets identified by J and R

- They both have height 1, but J has more elements, so we add R to J





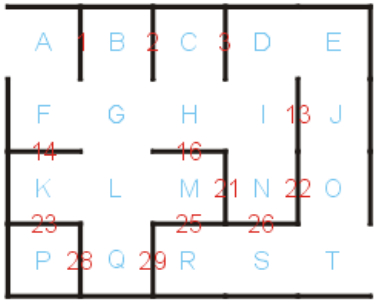


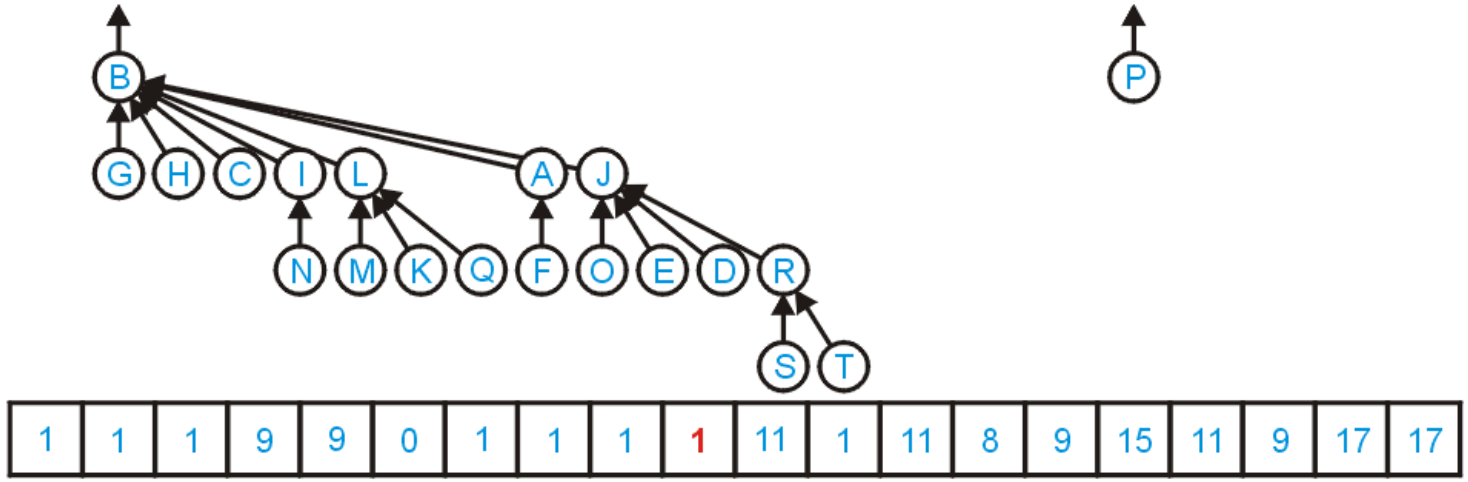
1	1	1	9	9	0	1	1	1	9	11	1	11	8	9	15	11	9	17	17

Application: Maze Generation

Selecting wall 8 joins sets identified by B and J

- They both have height 2 so we note that J has fewer nodes than B, so we add J to B

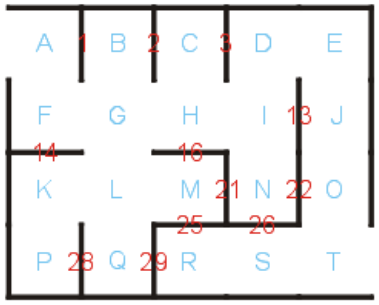


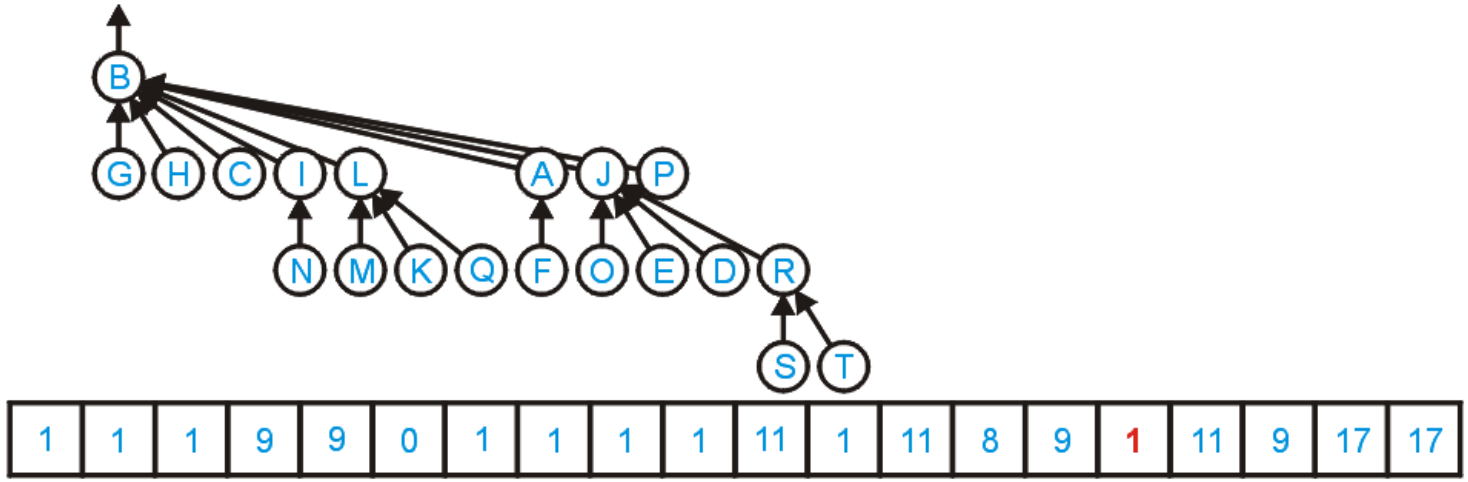


Application: Maze Generation

Finally we select wall 23 which joins the disjoint set P and the disjoint set identified by B

- P has height 0, so we attach it to B

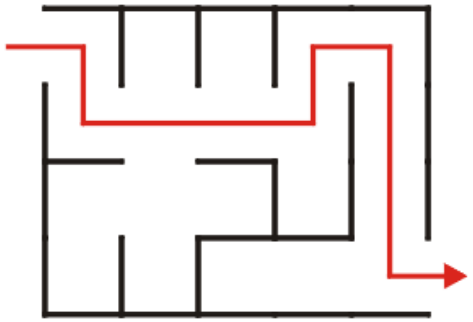




Application: Maze Generation

Thus we have a (rather trivial) maze where:

- there is one unique solution, and
- you can reach any square by a unique path from the starting point



Application: Maze Generation

You may also note that the average depth is 1.6 whereas the average depth of the worst-case disjoint tree is 2: