Adnar Lozano
06/09/15

Final

Due June 10, 2015 at 11:59 pm
Submit to marc.soriano@rcc.edu

1) 4 * 3 = 12
There are 12 different ways to go from LA to Temecula through Riverside.

2) Pigeonhole Principle k + 1. Therefore 4 +1 = 5
You will need to pull AT MOST 5 socks to guarantee a matching pair.
Answer is d) 5.

3) If there are 26 letters in the alphabet and 10 numbers (0-9)
AA### = 26 * 26 * 10 * 10 * 10 = 676,000
A total of 676,000 serial numbers can be generated from the given format

4) The source node is A, and the sink node is H.

5) $P(HHH) = 1 / 2^3 = 1 / 8 = 0.125$
The probability of flipping and getting 3 consecutive heads is 12.5%.

6) $P(HHH) = 0.51^3 = 0.132651$
Assuming we are using a weighted coin that gives head P(H) = 0.51, and tails P(T) = 0.49, the probability of flipping and getting 3 consecutive heads is 13.3%.

7) Assuming root node = height 0. The number of nodes (n) in a complete binary tree is $2^{h+1} - 1$, where h is the height of the tree.
Let h = 8
$n = 2^{8+1} - 1 = 2^9 - 1 = 511$
The maximum number of nodes that a tree with height 8 might have is 511.
Answer is e) 511.

8) (52/52) * (51/52) * (50/52) * (49/52) * (48/52) *
(47/52) * (46/52) * (45/52) * (44/52) * (43/52) = 0397133
The probability that will not draw any repeated cards is 39.71%
1 - 0.397133 = 0.602867
The probability that will draw at least one repeated card is 60.29%

9) a) 10^4 = 10,000 possible key codes.
b) 4! = 24 possible key codes.
c) 3*3*2*1 = 18 possible key codes.

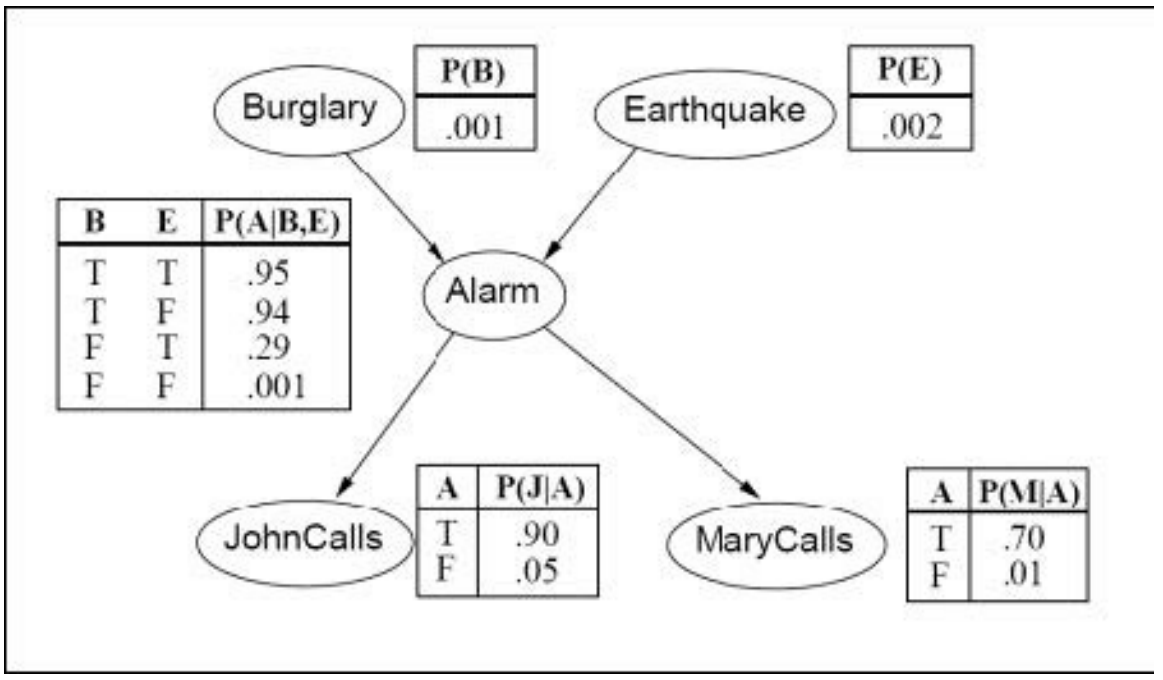
10) Assume you have a Mars rover that can move right a single unit with a probability of P(M) = 0.6 Given the following initial probability distribution at t=0

1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

What is the probability distribution after 5 steps? (i.e. t=5)

t = 0	t = 1	t = 2	t = 3	t = 4	t = 5		
0.01024	0.0768	0.2304	0.3456	0.2592	0.07776	0.0	0.0

11) Assume you are given the following Bayesian Network for an alarm system owned by John and Mary.



Determine the probably that either John or Mary will call the police with their dependence on the alarm marginalized out. (Hint: Determine P(A) first then apply to find P(J) + P(M))

$$P(A=t) = P(A \mid B \mid E) * P(B) * P(E) + P(A \mid B \mid \overline{E}) * P(B) * P(\overline{E}) +$$
$$P(A \mid \overline{B} \mid E) * P(\overline{B}) * P(E) + P(A \mid \overline{B} \mid \overline{E}) * P(\overline{B}) * P(\overline{E}) +$$
$$= (0.95)(0.001)(0.002) + (0.94)(0.001)(0.998) +$$
$$(0.29)(0.999)(0.002) + (0.001)(0.999)(0.998)$$
$$= \boxed{0.002516442}$$

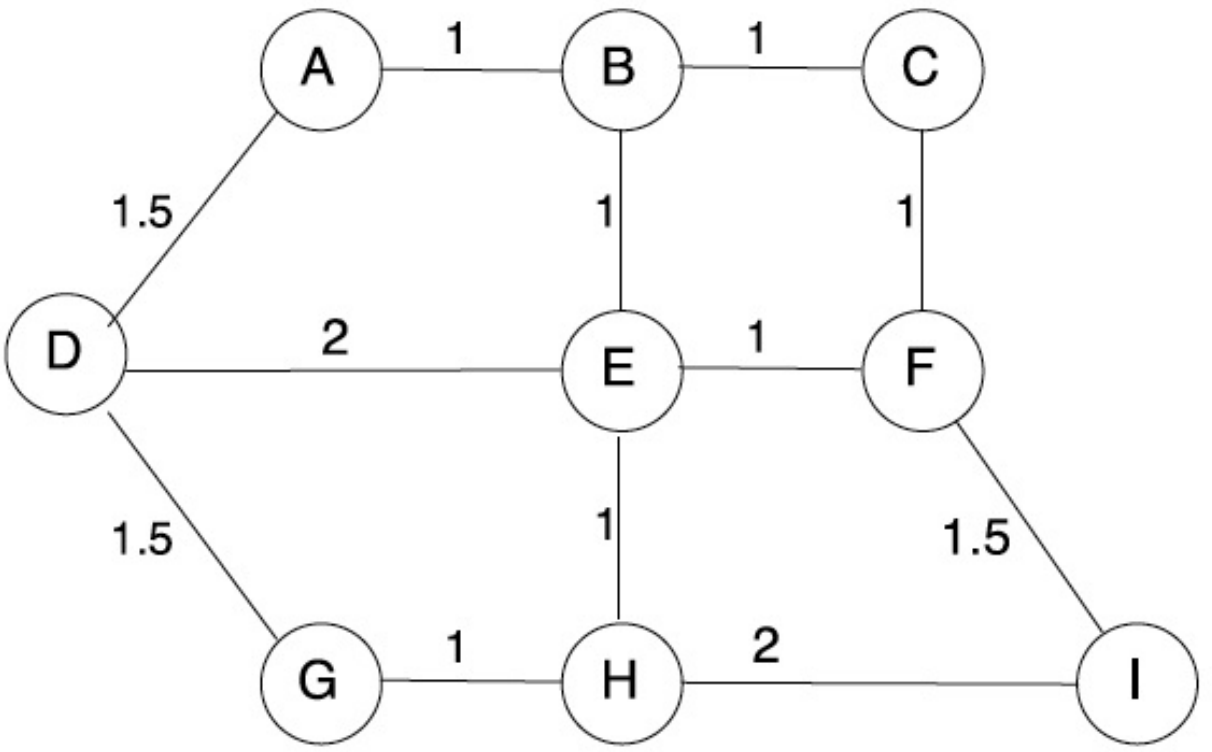
$$P(J=t) = P(J \mid A) * P(A) + P(J \mid \overline{A}) * P(\overline{A})$$
$$= (0.90)(0.0025) + (0.05)(0.9975)$$
$$= 0.00225 + 0.049875$$
$$= 0.052125$$

$$P(M=t) = P(M \mid A) * P(A) + P(M \mid \overline{A}) * P(\overline{A})$$
$$= (0.70)(0.0025) + (0.01)(0.9975)$$
$$= 0.00195 + 0.009975$$
$$= 0.011725$$

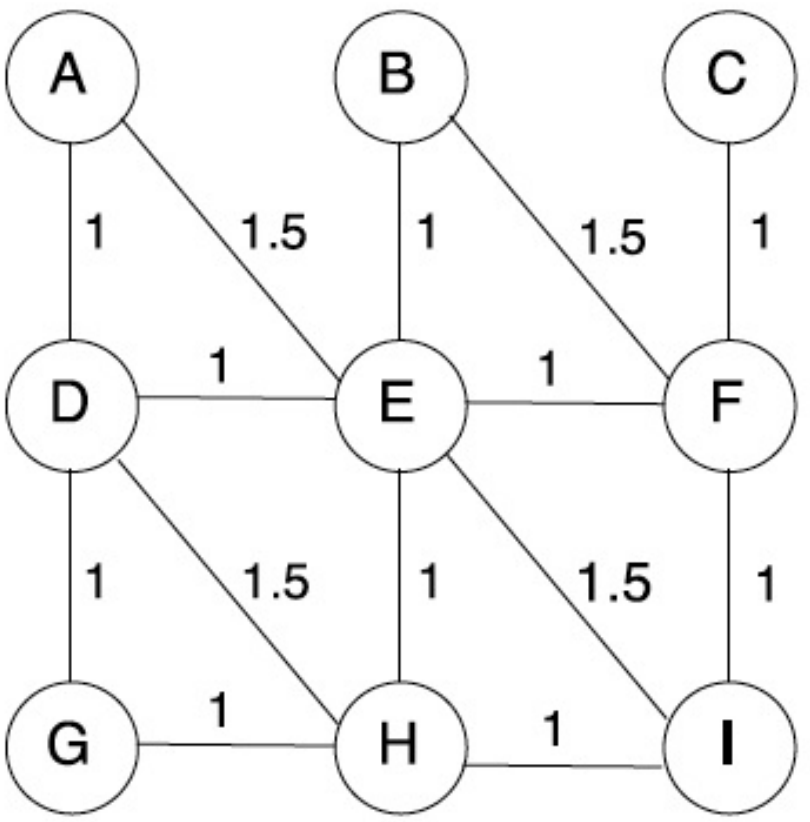
$$P(J) + P(M) = 0.052125 + 0.011725 = \boxed{0.06385}$$

The probability that either John or Mary will call the police is 6.3%

12) Draw and label a corresponding graph given the following weighted adjacency matrix (disconnected edges are labeled with ∞).

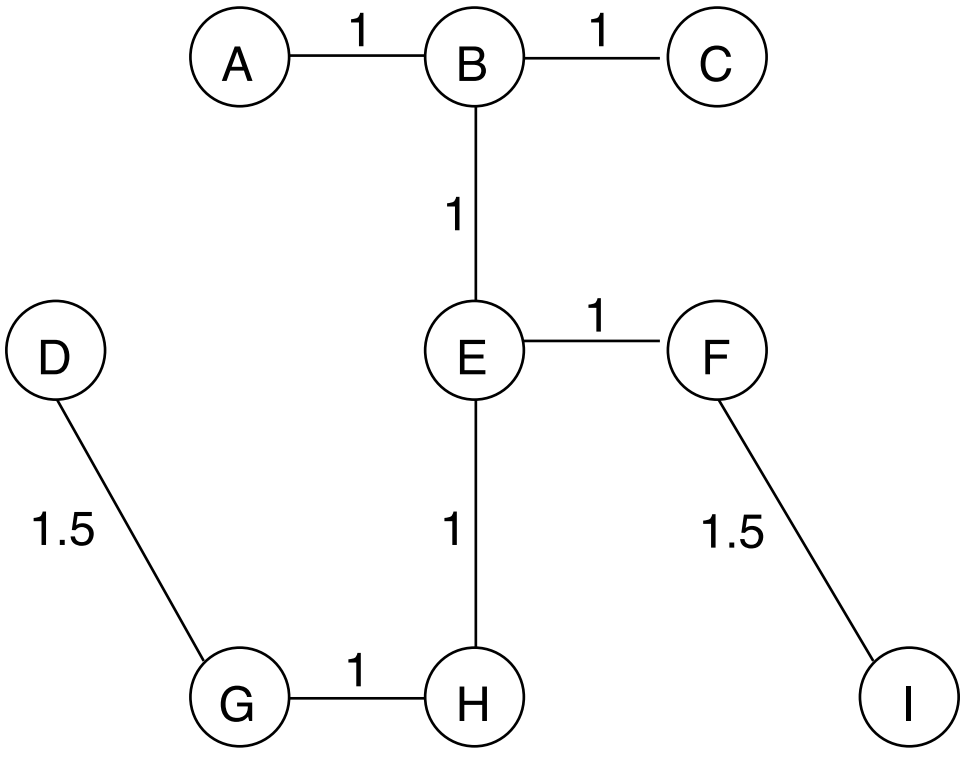


13) Draw and label a corresponding graph given the following weighted adjacency list

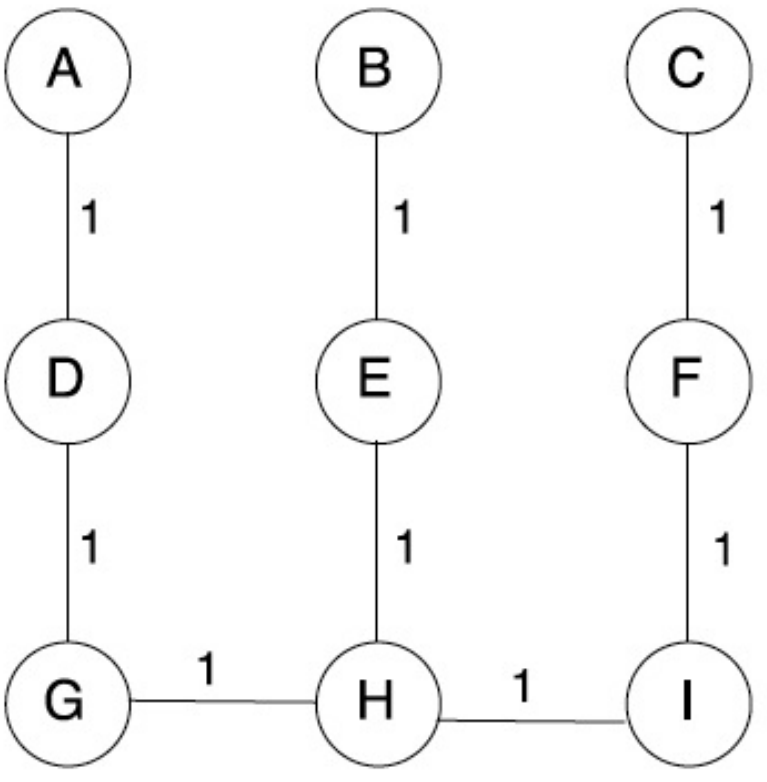


14) Determine the minimum spanning trees for the graphs in problem 12 and 13

Problem 12



Problem 13



15) a) Come up with an algorithm to find all sets of size 4 of unique numbers between 1 and 100 that sum up to 100.

```
// Adnar Lozano
// CIS-7 Discrete Structures
// Problem 15 a
#include <iostream>
using namespace std;
int main(){
    for (int i = 1; i < 100; ++i)
      for (int j = i; j < 100; ++j)
        for (int k = j; k < 100; ++k)
            for(int m = k; m < 100; ++m)
                if ( i != j && j != k && k != m)
                    if(i+j+k+m ==100)
                        cout << i <<" + "<< j <<" + "<< k <<" + "<< m<< " = 100"<<endl;
    return 0;
}
```

b) Come up with an algorithm to find all sets of non-repeating numbers between 1 and 100 that sum up to 100.

```
// Adnar Lozano
// CIS-7 Discrete Structures
// Problem 15 b
#include <iostream>
#include <list>
using namespace std;
int counts = 1;
void recSum(list<int> numbers, int target, list<int> partialSum){
    int sum = 0;
        for (list<int>::iterator list = partialSum.begin(); list != partialSum.end(); list++)
            sum += *list;
        if(sum == target)
        {
            cout << "#" << counts << ": ";

                for (list<int>::iterator list = partialSum.begin(); list != partialSum.end(); list++)
                    cout << *list << " ";
                cout << "= 100.\n";
                counts++;
        }
        if(sum >= target)
            return;
        int num;
        for (list<int>::iterator i = numbers.begin(); i != numbers.end(); i++)
        {
            num = *i;
           list<int> remaining;
            for(list<int>::iterator j = i; j != numbers.end(); j++)
            {
                if(j == i)
                continue;
                remaining.push_back(*j);
            }
            list<int> partialLst = partialSum;
            partialLst.push_back(num);
            recSum(remaining,target,partialLst);
        }
}
void summation(list<int> numbers,int target)
{
    recSum(numbers,target,list<int>());
}
int main()
{
    list<int> set;
    for(int i = 1; i < 100; i++)
    {
    set.push_back (i);
    }
    int total = 100;
    summation(set, total);
    return 0;
}
```

16) Develop a recursive algorithm to determine the height of a binary tree. Each node might have a left or right child, they will be assigned to NULL if they do not exist. (8 points)

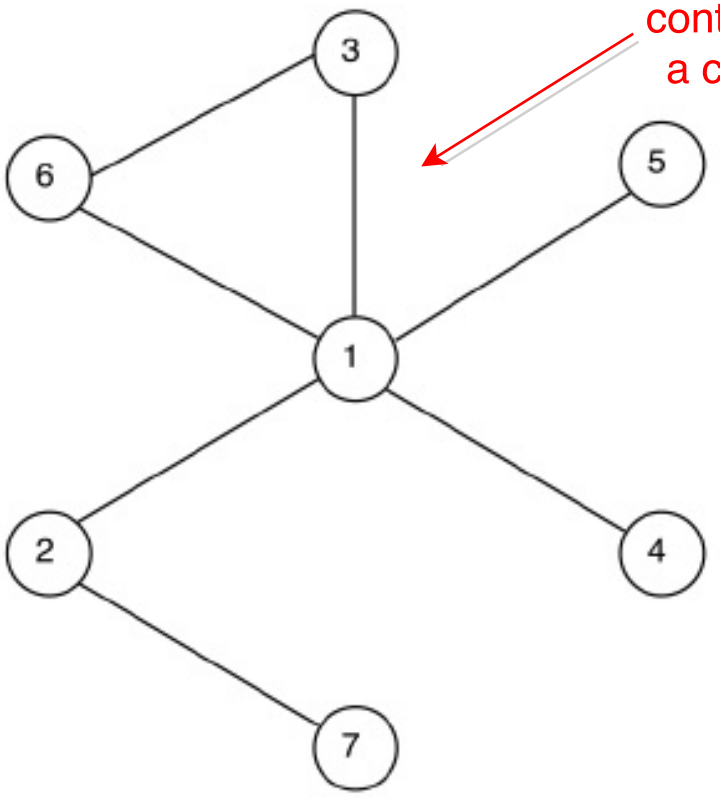
```
1   // Adnar Lozano
2   // CIS-7 Discrete Structures
3   // Problem 16
4   struct Node
5   {
6       Node* left;
7       Node* right;
8   };
9   int height(node* n)
10  {
11      if (n == NULL)
12          return 0;
13      else
14      {
15          int leftH = height(n->left);
16          int rightH = height(n->right);
17          if(leftH > rightH)
18              return(leftH+1);
19          else return(rightH+1);
20      }
21  }
```

17) Write an algorithm that given an undirected adjacency matrix determines if the resulting graph is a tree or not. (Alternative: determine if the graph contains any cycles) See the given files for test examples.

```
1   // Adnar Lozano
2   // CIS-7 Discrete Structures
3   // Problem 17
4   #include <iostream>
5   #include <fstream>
6   #include <string>
7   using namespace std;
8   int values;
9   int** results;
10  int visitedNode[7] = {0};
11  int edges = 0;
12  int** LoadMatrix(string filename){
13      ifstream in;
14      in.open(filename);
15      in >> values;
16      int **result;
17      result = new int *[values];
18      for ( int i = 0; i < values; i++ )
19            result[i] = new int[values];
20      for ( int i = 0; i < values; i++ ) {
21        for ( int j = 0; j < values; j++ )
22          in >> result[i][j]; }
23      return result;
24  }
25  void isGraph(int v){
26      visitedNode[v] = true;
27      for(int i = 0; i < values; i++) {
28          if(!visitedNode[i]&&results[v][i]== true)
29              isGraph(i); }
30      for(int i = 0; i < values; i++) {
31              if (visitedNode[i] == false) {
32                  cout << "This graph is NOT a tree.\n";
33                          exit(0); } }
34      if( edges == values-1)
35          cout << "This graph Is a tree\n";
36      else
37          cout << "This graph is Not a tree.\n";
38      exit(0);
39  }
40  int main(){
41      string filename;
42      cout << "Enter the filename: ";
43      cin >> filename;
44      results = LoadMatrix(filename);
45      for ( int i = 0; i < values; i++ ) {
46        for ( int j = 0; j < values; j++ ) {
47          if(results[i][j] == 1) {
48              edges++; } } }
49      edges = edges / 2;
50      isGraph(0);
51      return 0;
52  }
```

17) Is it a Tree?

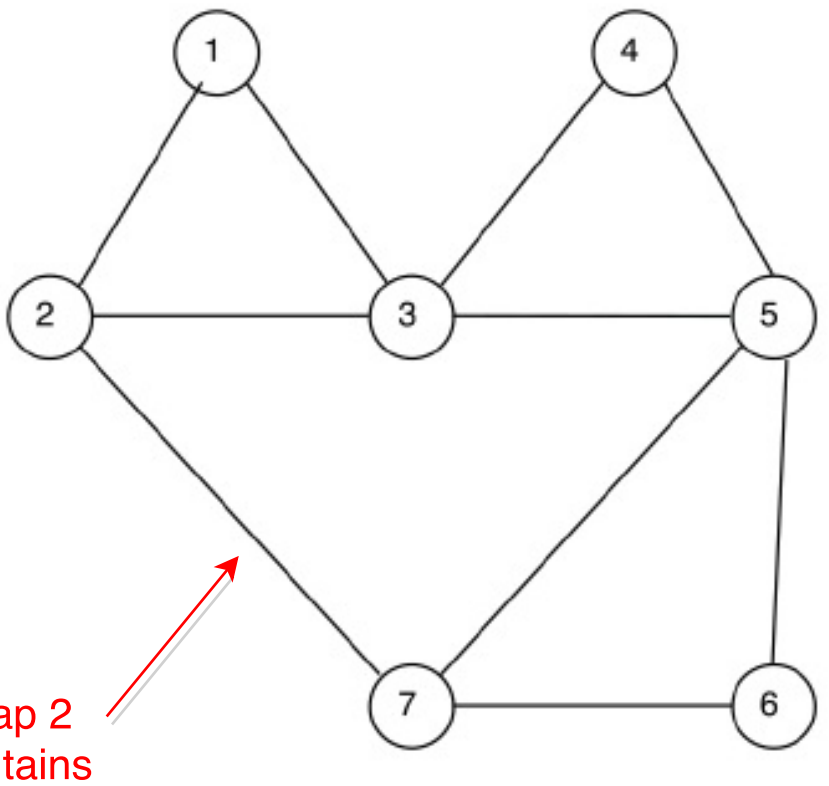
Map 1



Map 1 contains a cycle

Map 1 IS NOT a tree

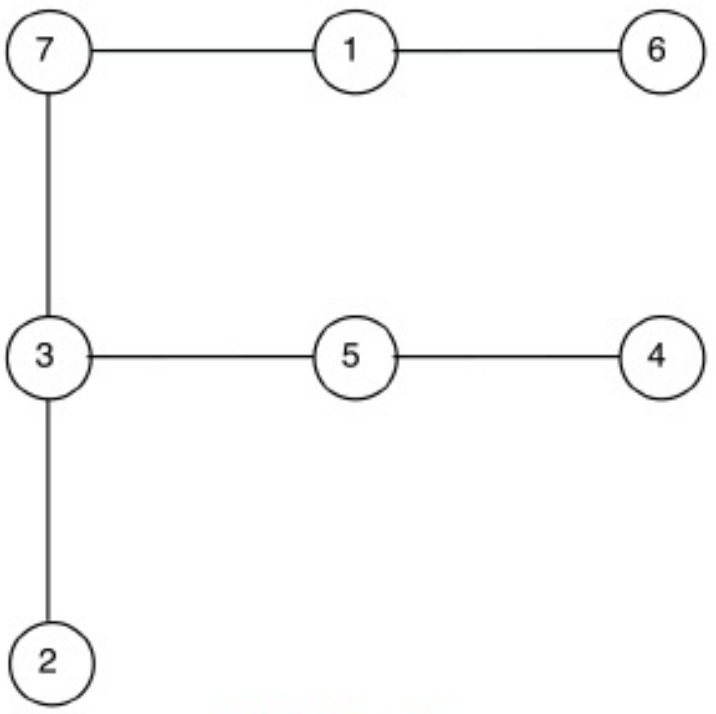
Map 2



Map 2 contains many cycles

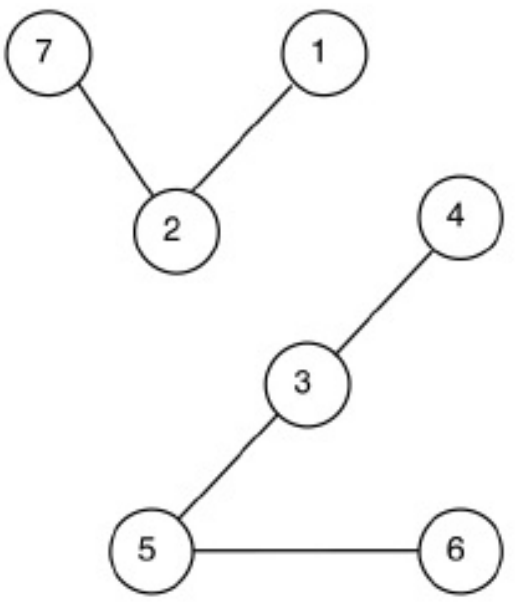
Map 2 IS NOT a tree

Map 3



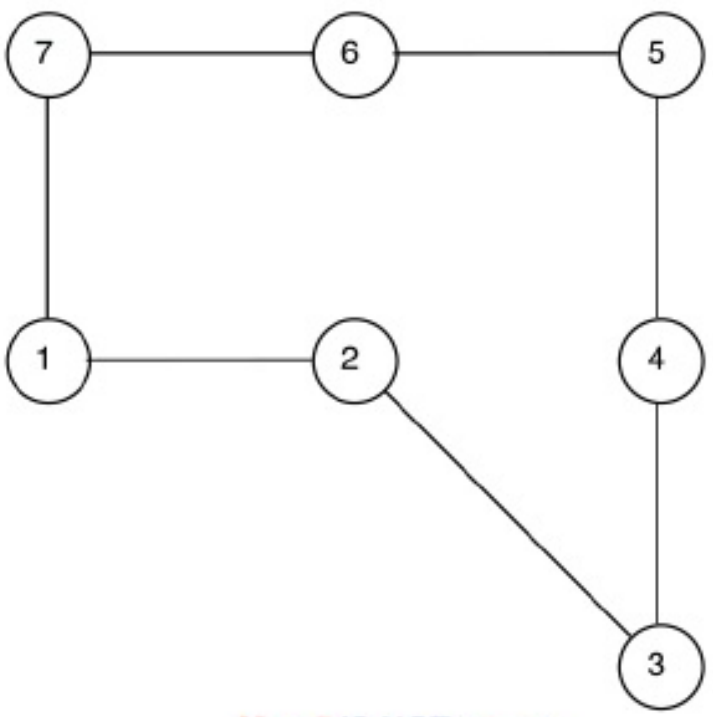
Map 3 IS a tree

Map 4



Map 4 IS NOT a tree

Map 5



Map 5 contains a cycle

Map 5 IS NOT a tree