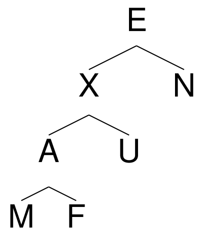
DSA Homework #4

4.1 Trees

1.



2. In the tree T, there are n nodes $a_1, a_2, a_3, ..., a_n$.
   Claim:for every node $a_i(0 < i < n)$, $f(a_i) \leq 2^n - 1$.

   - For $n = 1$:
     The only element must be the root element. From the definition given in page 295, its level numbering is 1.
     Because $1 \leq 2^1 - 1$, the claim is true for $n = 1$.

   - Assume for $n = k$, $f(a_i) \leq 2^n - 1$ $(0 < i < n)$

   - For $n = k + 1$,

$$f(a_{k+1}) = \begin{cases} f(a_{k+1}) = 2f(a_i) & \text{if v is the left child of node } a_i \\ f(a_{k+1}) = 2f(a_i) + 1 & \text{if v is the right child of node } a_i \end{cases}$$

$$\begin{aligned} a_{k+1} &\leq 2f(a_i) + 1 \\ &\leq 2(2^n - 1) + 1 \\ &= 2^{n+1} - 1 \end{aligned}$$

     Therefore, the claim is true when $k = n + 1$.

   From mathematical induction, it prove that for every node $a_i(0 < i < n)$, $f(a_i) \leq 2^n - 1$.
   That is, for every node v of T , $f(v) \leq 2^n - 1$.

3. Because in a postorder travelsal, the node v is traveled before all its ascendant and after all its descendant. In a preorder travelsal, the node v is traveled before all its descendant and after all its ascendant. Therefore, we can derive the formula:
   $post(v) = pre(v) - depth(v) + desc(v) + 1$

4.

```
function printTree( Node*node , string indent ){
  if( node -> isLeave() ){
    printLine( indent + node -> value );
  }else{
    printLine( node -> value + "(" );
    for( childNode in node ){
        printTree( childNode , indent + "    " );
    }
    printLine( ")" );
  }
}
```

4.2 Decision Tree

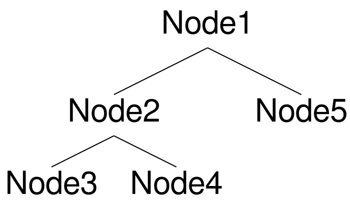
1.

```
nYesRight = nTotalYes; nNoRight = nTotalYes;
nYesLeft = 0; nNo = 0;
double bestThreshold;
double minConfusion = 1;
for( i = 0 to m - 2 ){
  //split those examples to left and right subsets
  if( belongExampleDesision( v_i, ) == YES ){
    nYesLeft += 1;
    nYesRight -= 1;
  }else{
    nNoLeft += 1;
    nNoRight -=1;
  }
  double confusion = calculateConfusion( nYesLeft, nNoLeft, nYesRight, nNoRight );
  if( confusion < minConfusion ){
    bestThreshold = (v_i, + v_{i+1} ) / 2
  }
}
```

2.

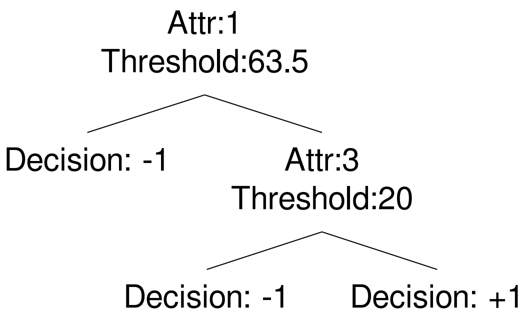
3. The tree is composed with nodes. And each node contain two pointer to its child, the threshold, and the attribute the threshold belongs to. Also, in case the node is a leave, so the node also have a variable storing decision.

```
struct Node{
  //Meaningful only when node is not leave
  Node*left;
  Node*right;
  Threshold threshold;
  Attr attr;
  //Meaningful only when node is leave
  short decision = 0;
};
```



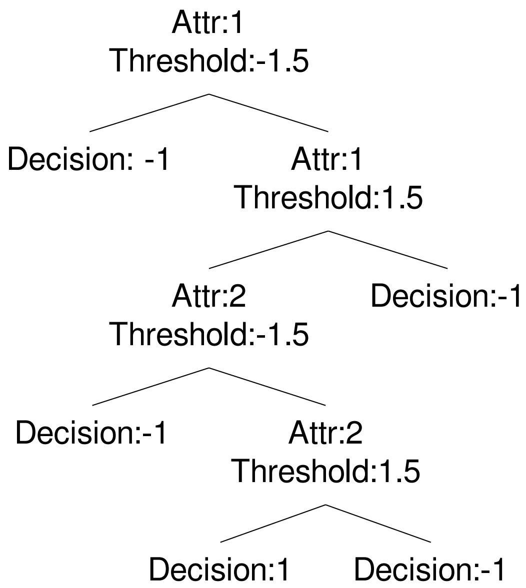
0x1	left = 0x6
0x2	right = 0x16
0x3	threshold = 3
0x4	attr = 3
0x5	decision = 0
0x6	left = 0xB
0x7	right = 0x11
0x8	threshold = 2.7
0x9	attr = 2
0xA	decision = 0
0xB	left = NULL
0xC	right = NULL
0xD	threshold = 0
0xF	attr = 0
0x10	decision = +1
0x11	left = NULL
0x12	right = NULL
0x13	threshold = 0
0x14	attr = 0
0x15	decision = -1
0x16	left = NULL
0x17	right = NULL
0x18	threshold = 0
0x19	attr = 0
0x1A	decision = -1

4.

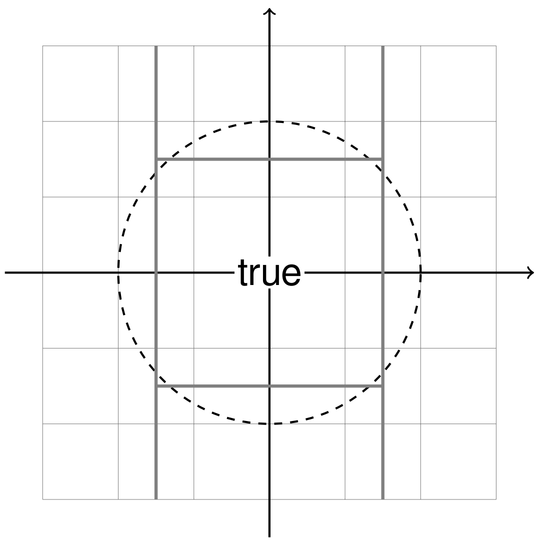


5. I teach my tree with the data set that whether or not the ponits from $(3,3)$ to $(-3,-3)$ is within the circle $x^2 + y^2 \leq 4$

```
-1 1:3 2:3
-1 1:3 2:2
-1 1:3 2:1
-1 1:3 2:0
-1 1:3 2:-1
-1 1:3 2:-2
-1 1:3 2:-3
-1 1:2 2:3
...
+1 1:0 2:1
+1 1:0 2:0
+1 1:0 2:-1
...
-1 1:-2 2:-3
-1 1:-3 2:3
-1 1:-3 2:2
-1 1:-3 2:1
-1 1:-3 2:0
-1 1:-3 2:-1
-1 1:-3 2:-2
-1 1:-3 2:-3
```



The tree split those examples based on x first, and it split the graph into 3 regions. And then, it split the center region into 3 regions based on y.

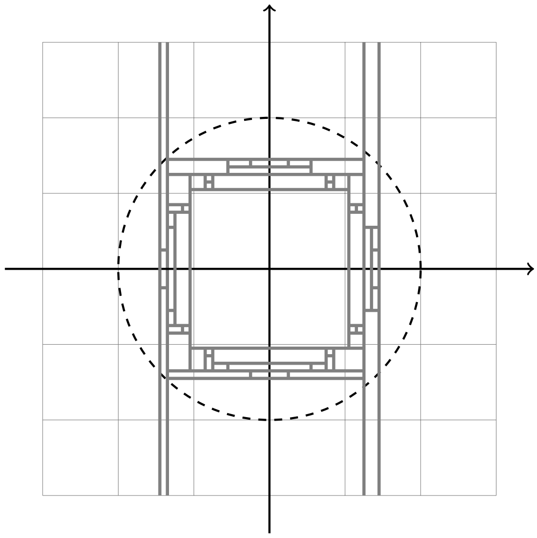


My code genarete test data set:

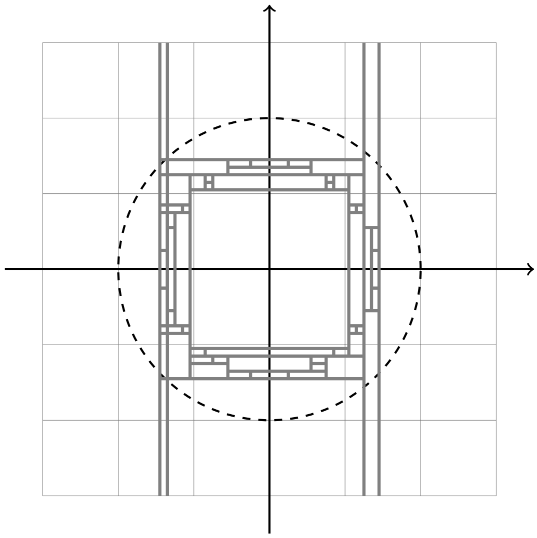
```
#define RADIUS 2
#define BOUND 3
#define PRECISE 1.0
int main(){
        for( int i = BOUND * PRECISE ; i >= -BOUND * PRECISE ; --i ){
                for( int j = BOUND * PRECISE ;
                      j >= -BOUND * PRECISE ; --j ){
                        var x = i / PRECISE ;
                        var y = j / PRECISE ;
                        std::cout << (inCircle(x ,y ,RADIUS )?"+1":"-1")
                                  << " 1:" << x
                                  << " 2:" << y << std::endl;
                }
        }
}
```

4.3 (Play for fun)

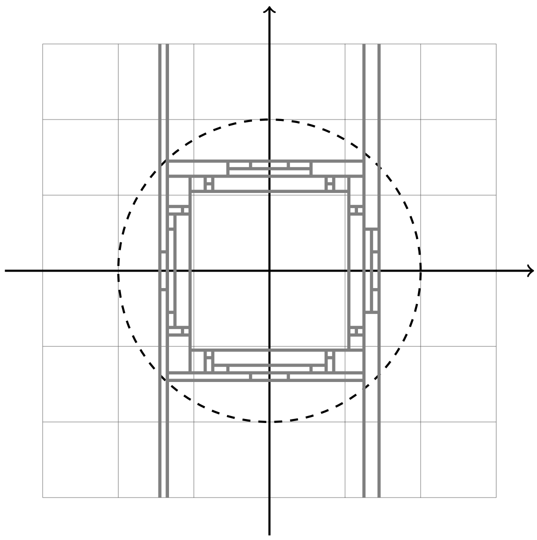
If I change the PRECISE in the test data generator program to 10.0, the graph becomes ->.
I guess if I increase BOUND, my tree will more like the circle.



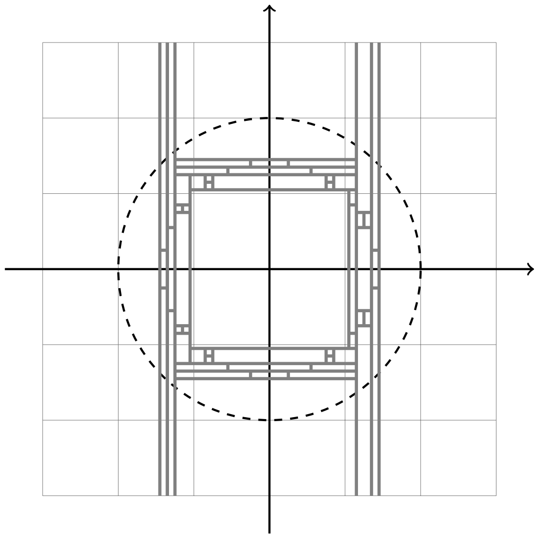
However, when BOUND = 100, the way my tree split is still not like the circle.



I try to reduce the BOUND to 2, with the same precision.



Do an experiment, reduce the BOUND to 1.5



Finally BOUND = 2, PRECISE = 50;