

2. TREE DATA STRUCTURES

Topics:

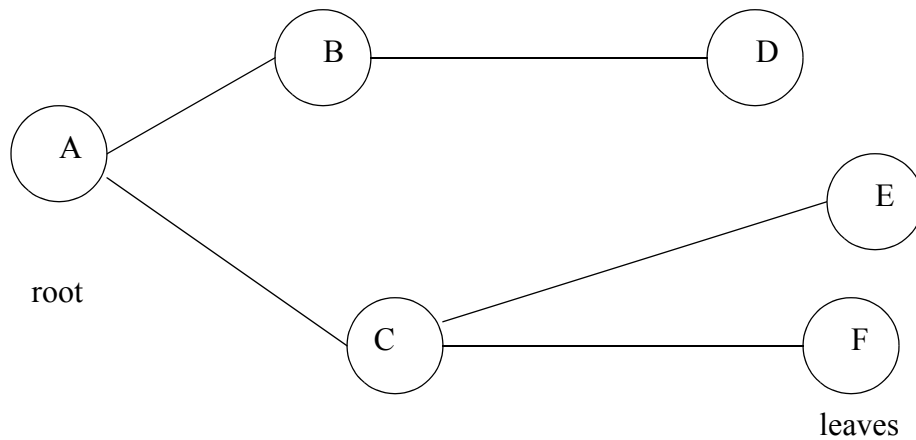
-

Objectives:

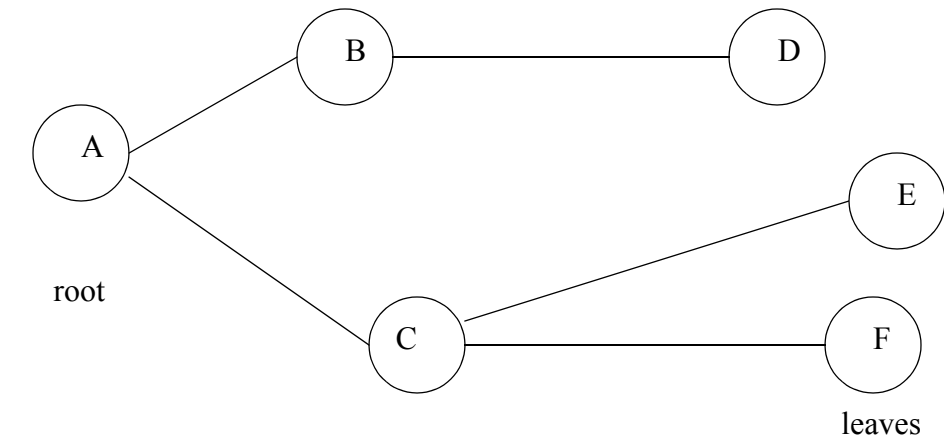
-

2.1 Introduction

- Trees are a special case of a graph data structure. The connections radiate out from a single root without cross connections.
- The tree has nodes (shown with circles) that are connected with branches. Each node will have a parent node (except for the root) and may have multiple child nodes.



- Consider the graph below.



Node #	Node Name	Parent	Child
1	A		
2	B	1	2
3	C	1	3
4	D	2	4
5	E	3	5
6	F	3	6

- Representing a tree in Scilab

```

function foo = find_index(_list, name) // a function to find list numbers given names
    foo = -1; // use -1 to indicate no match found yet
    A = size(_list); // find the rows and columns in the list
    cnt = A(1, 1); // get the rows in the list
    for i = 1 : cnt, // loop through the rows
        if name == _list(i) then // look for a name match
            foo = i; // record the matching row number
            break; // no point continuing the for loop
        end,
    end
end
if foo == -1 then mprintf("ERROR: list name %s not found\n", name); end
endfunction

node_names = ["A" ; "B" ; "C" ; "D" ; "E" ; "F"]; // define the nodes

function foo = node_number(name) // a function to find node numbers given names
    foo = find_index(node_names, name);
endfunction

branch = [node_number("A"), node_number("B")];
branch = [branch ; [node_number("A"), node_number("C")]];
branch = [branch ; [node_number("B"), node_number("D")]];
branch = [branch ; [node_number("C"), node_number("E")]];
branch = [branch ; [node_number("C"), node_number("F")]];
  
```

• Representing a graph in C

```

int find_number(char list[][0], char *name){
    int i;
    if(name == NULL) return -1;
    for(i = 0; list[i][0] != NULL; i++){
        if(strcmp(name, list[i]) == 0){
            return i;
        }
    }
    printf("ERROR: search name %s not found \n", name);
    return -1;
}

char node_names[][0] = {"A"}, {"B"}, {"C"}, {"D"}, {"E"}, {"F"}, NULL};
int node_number(char *name){ return find_number(node_names, name);}

#define MAX_BRANCHES 10
int branch_cnt = 0;
int branch[MAX_BRANCHES][2];

void add_branch(int parent, int child){
    branch[branch_cnt][0] = node_number(parent);
    branch[branch_cnt][1] = node_number(child);
    branch_cnt++;
}

int main(){
    add_branch("A", "B");
    add_branch("A", "C");
    add_branch("B", "D");
    add_branch("C", "E");
    add_branch("C", "F");

    return 0;
}

```

2.3 Applications

2.3.1 Precedence Identification

- To determine a precedence we add a precedence count for each node. The basic check is,
 1. Set all the precedence values to 1 (lowest precedence)
 2. Cycle through and check each branch. For each branch ensure that the child vertex has a precedence that is one greater than the parent vertex.

- Consider the previous example.

```

function foo = find_index(_list, name) // a function to find list numbers given names
    foo = -1; // use -1 to indicate no match found yet
    A = size(_list); // find the rows and columns in the list
    cnt = A(1, 1); // get the rows in the list
    for i = 1 : cnt, // loop through the rows
        if name == _list(i) then // look for a name match
            foo = i; // record the matching row number
            break; // no point continuing the for loop
        end,
    end
    if foo == -1 then mprintf("ERROR: list name %s not found\n", name); end
endfunction

node_names = ["A" ; "B" ; "C" ; "D" ; "E" ; "F"]; // define the nodes
node_cnt = 6; // get the rows in the list

function foo = node_number(name) // a function to find node numbers given names
    foo = find_index(node_names, name);
endfunction

branch = [node_number("A"), node_number("B")];
branch = [branch ; [node_number("A"), node_number("C")]];
branch = [branch ; [node_number("B"), node_number("D")]];
branch = [branch ; [node_number("C"), node_number("E")]];
branch = [branch ; [node_number("C"), node_number("F")]];
branch_cnt = 5;

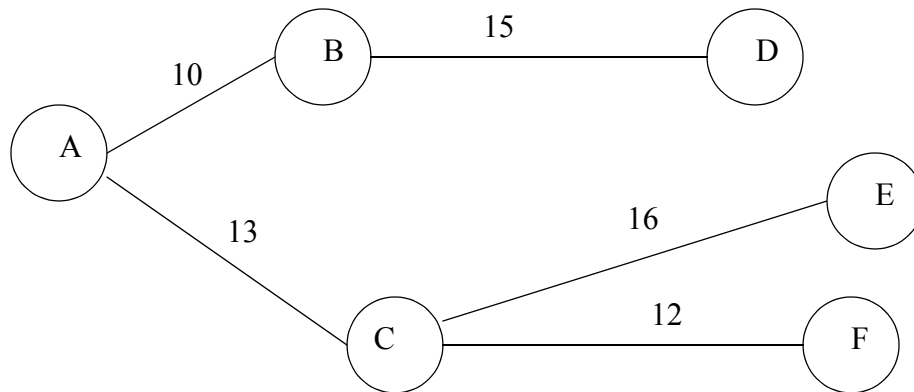
for i=1:node_cnt, // set all precedence values to 1
    precedence(i) = 1;
end

for i=1:node_cnt-1, // loop through and update precedence values
    for j=1:branch_cnt,
        if precedence(branch(j,1)) >= precedence(branch(j,2)) then
            precedence(branch(j,2)) = precedence(branch(j,1)) + 1;
        end
    end
end
end

```

2.3.2 Tree Searching

- Assume that each branch has a cost associated. We can then find the cost of each of the nodes.



- Identifying the vertex costs.

```

function foo = find_index(_list, name) // a function to find list numbers given names
    foo = -1; // use -1 to indicate no match found yet
    A = size(_list); // find the rows and columns in the list
    cnt = A(1, 1); // get the rows in the list
    for i = 1 : cnt, // loop through the rows
        if name == _list(i) then // look for a name match
            foo = i; // record the matching row number
            break; // no point continuing the for loop
        end,
    end
    if foo == -1 then mprintf("ERROR: list name %s not found\n", name); end
endfunction

node_names = ["A" ; "B" ; "C" ; "D" ; "E" ; "F"]; // define the nodes
node_cnt = 6; // get the rows in the list

function foo = node_number(name) // a function to find node numbers given names
    foo = find_index(node_names, name);
endfunction

branch = [node_number("A"), node_number("B"), 10];
branch = [branch ; [node_number("A"), node_number("C")], 13];
branch = [branch ; [node_number("B"), node_number("D")], 15];
branch = [branch ; [node_number("C"), node_number("E")], 16];
branch = [branch ; [node_number("C"), node_number("F")], 12];
branch_cnt = 5;

for i=1:node_cnt, // set all node cost values to 0
    cost(i) = 0;
end

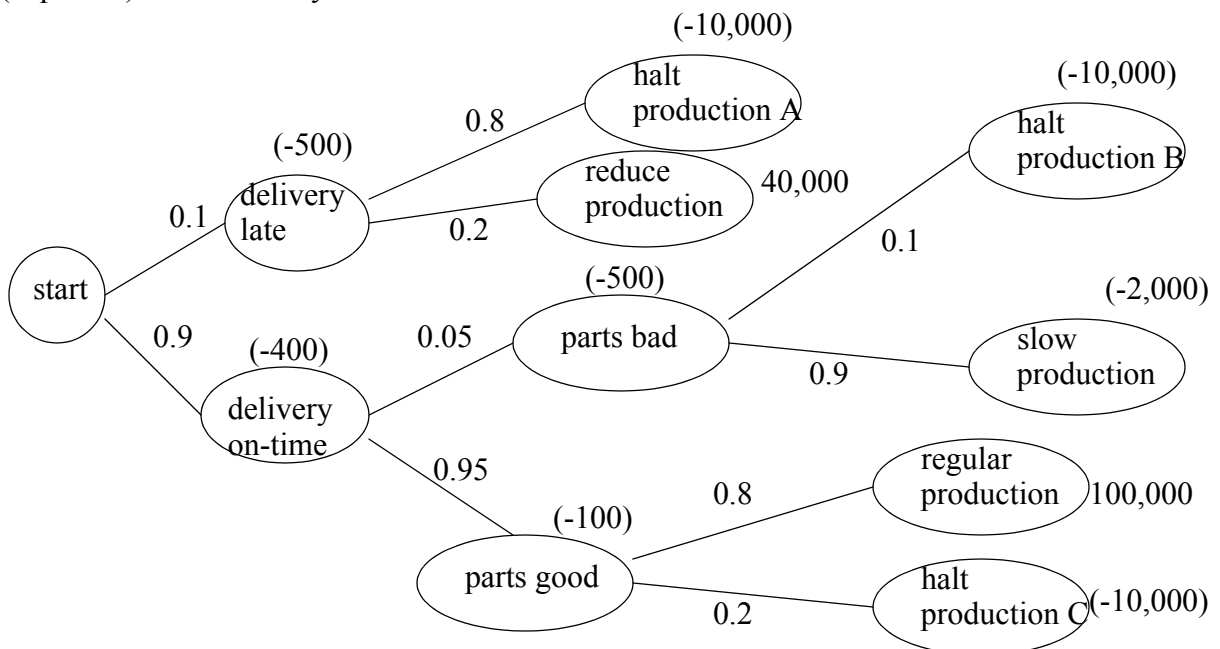
for i=1:node_cnt-1, // loop through and update cost values
    for j=1:branch_cnt,
        cost(branch(j, 2)) = cost(branch(j, 1)) + branch(j, 3);
    end
end
  
```

1.4 Summary

•

1.5 Problems

1. 1. Consider the following probability tree for a single day in a production facility. There is a 10% chance that 'delivery late' will occur, and an 90% chance that 'delivery on-time' will occur. The values on the branches indicate the probability of an event occurring. For example, if there is 'delivery on-time' there will be a net cost of \$400 to process the incoming parts. Write a program that will use the probabilities and costs to calculate the effective income (expenses) for the facility.



(ans. 66046)

1.6 Challenge Problems