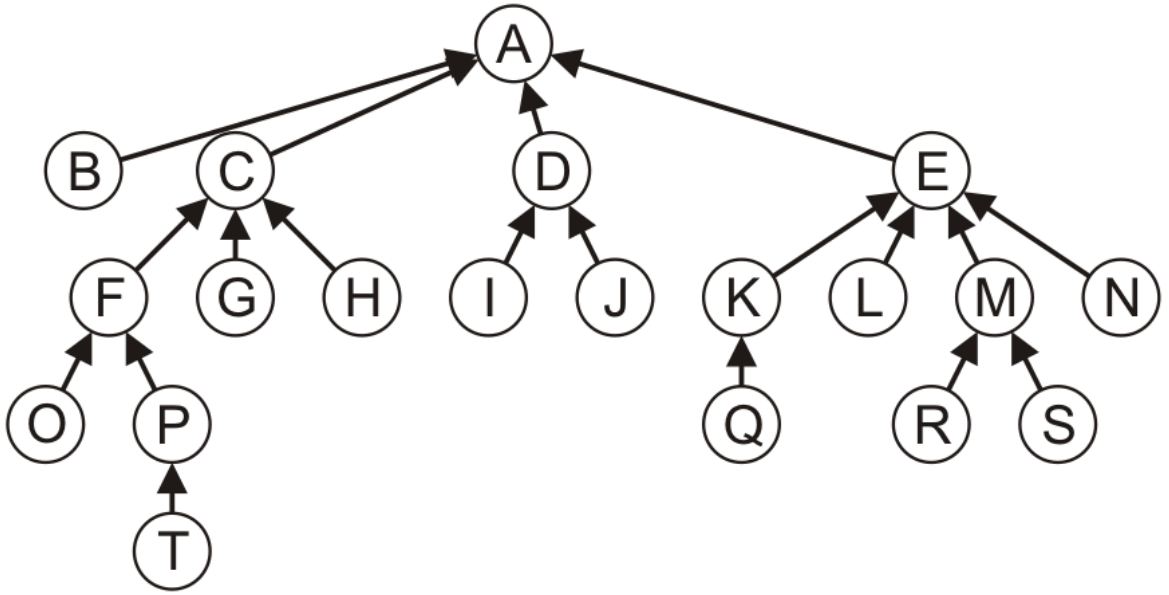
Parental trees and forest

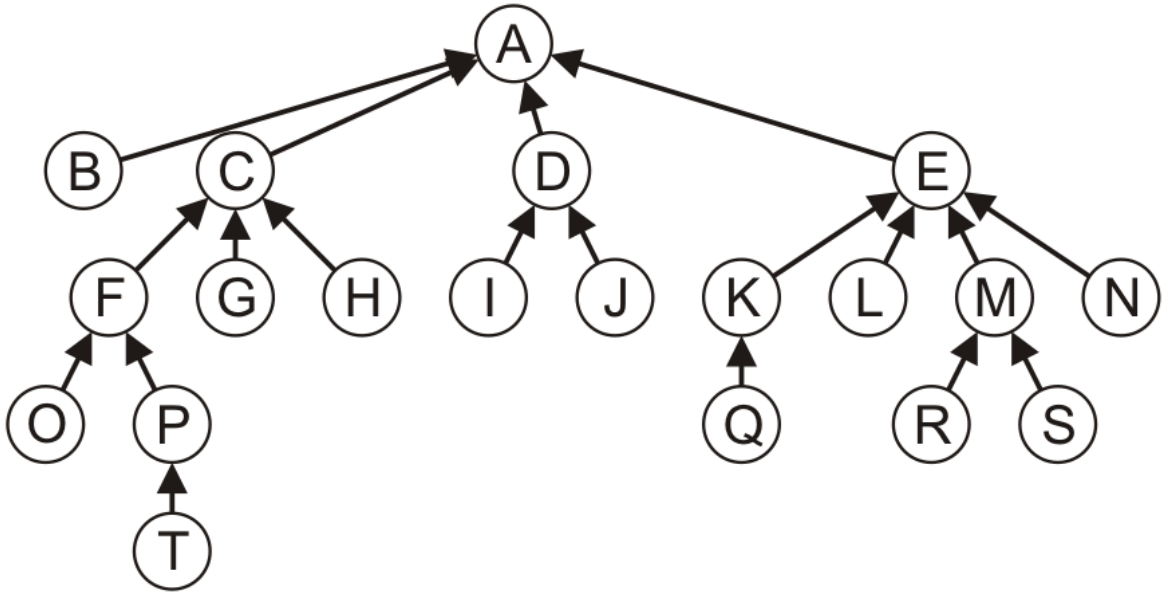
Definition

A *parental tree* is a tree where each node only keeps a reference to its parent node



Definition

This requires significantly less memory than our general tree structure, as no data structure is required to track the children



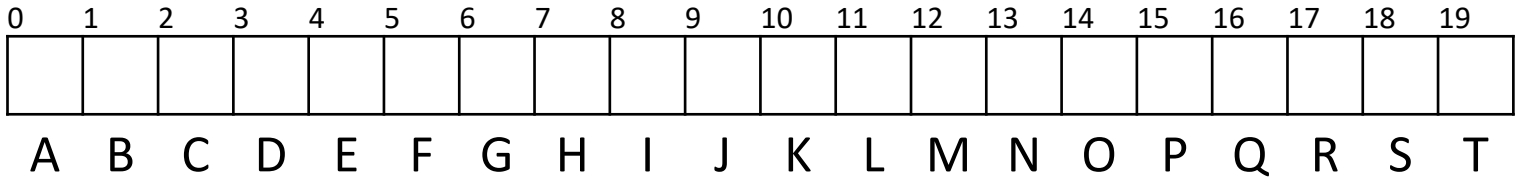
Implementation

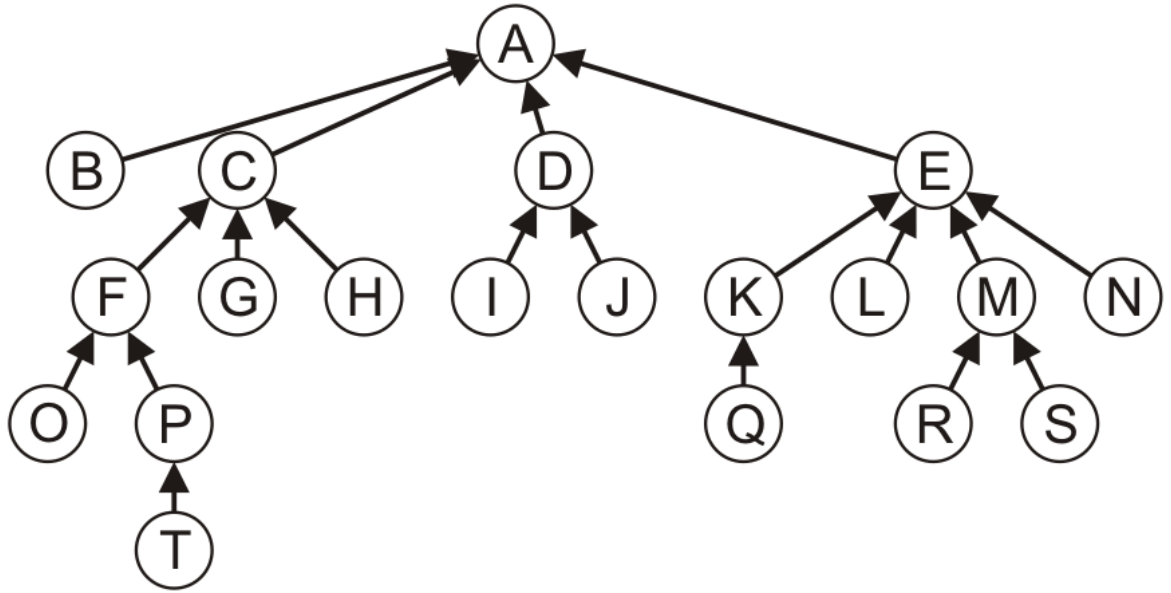
A naïve implementation may also be node based:

```
template <typename Type>
class Parental_tree {
    private:
        Type element;
        Parental_tree *parent;
    public:
        // ...
};
```

Implementation

Instead, generate an array of size n and associate each entry with a node in the tree



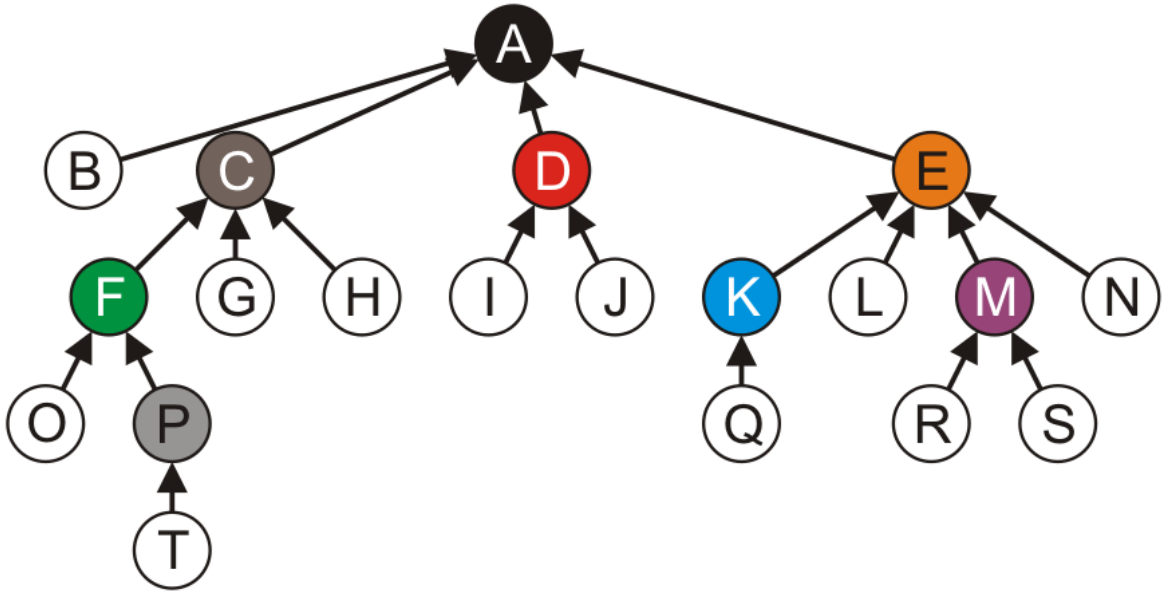


Implementation

Store the index of the parent in each node

- The root node, wherever it is, points to itself

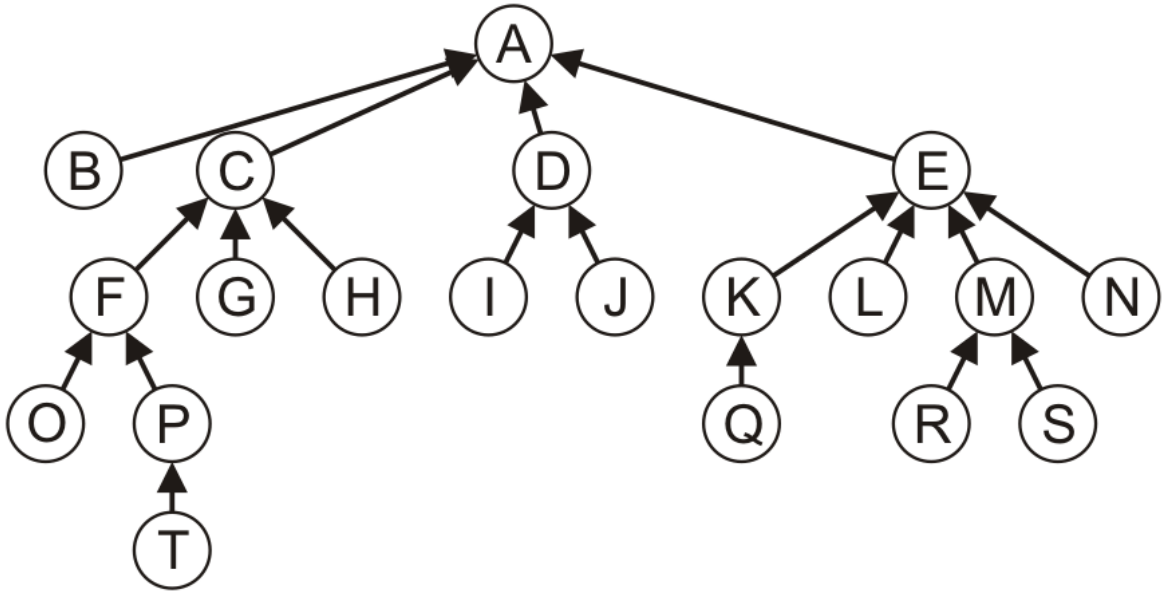
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	2	2	2	3	3	4	4	4	4	5	5	10	12	12	15
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T



Implementation

The memory requirements are quite small relative to our node-based implementation

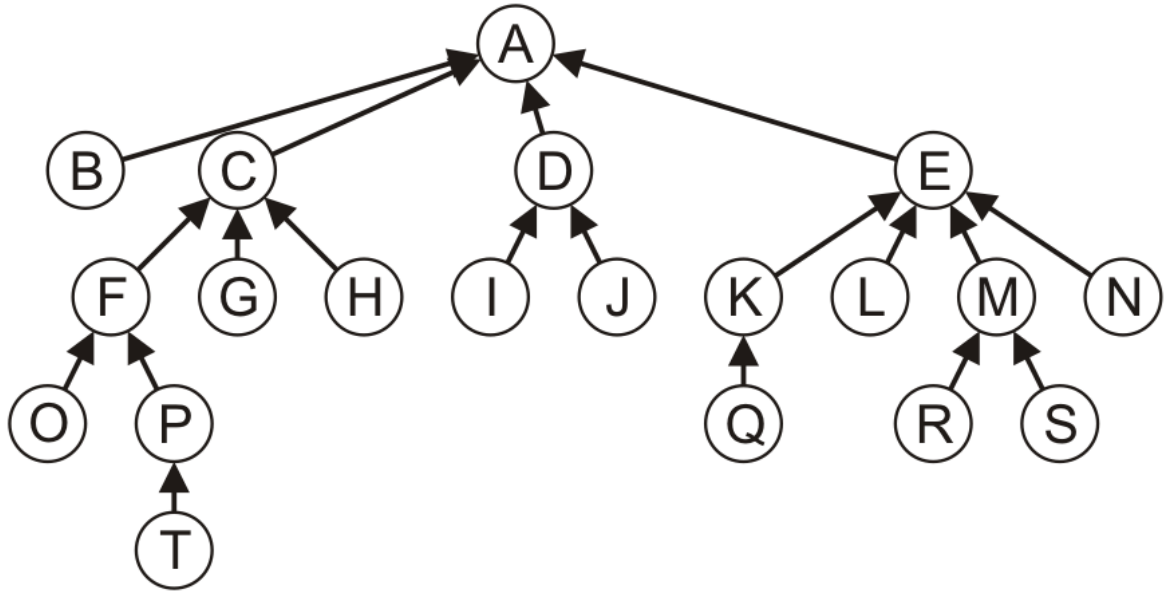
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	2	2	2	3	3	4	4	4	4	5	5	10	12	12	15
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T



Implementation

In a tree, only one node will point to itself

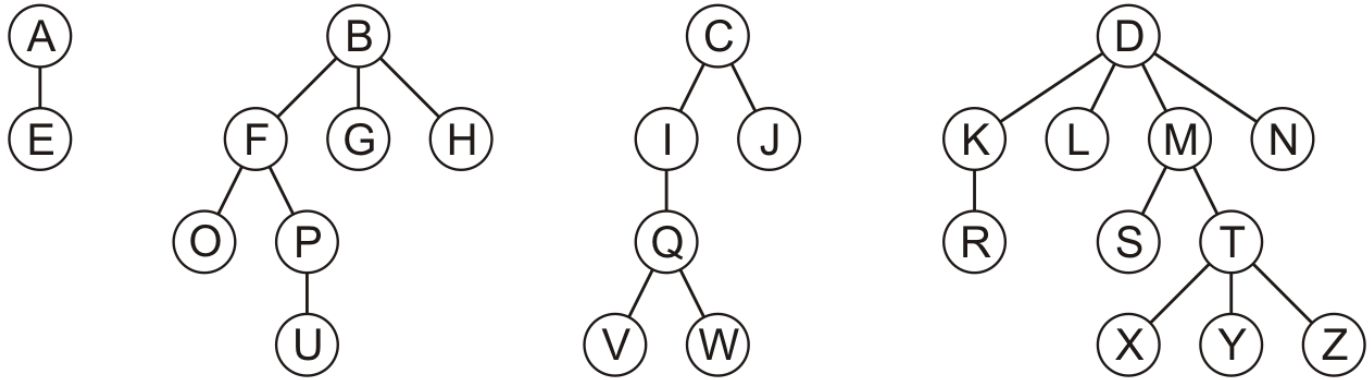
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	2	2	2	3	3	4	4	4	4	5	5	10	12	12	15
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T



Definition

A rooted forest is a data structure that is a collection of disjoint rooted trees

- A forest can be used to store the previously described relation



Definition

Note that:

- Any tree can be converted into a forest by removing the root node
- Any forest can be converted into a tree by adding a root node that has the roots of all the trees in the forest as children

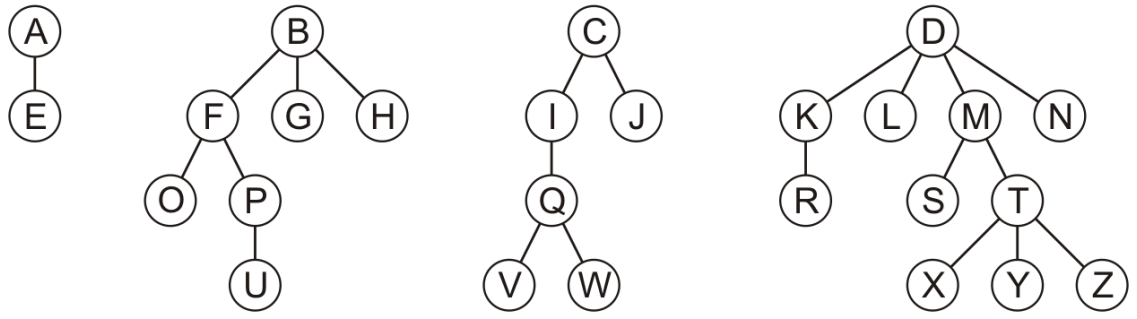
Implementation

Using the `Simple_tree` structure, one could simply keep a linked list of trees

```
Single_list<Simple_tree *> list;
```

Using the parental tree structure, one could remove the restriction that only one entry stores itself

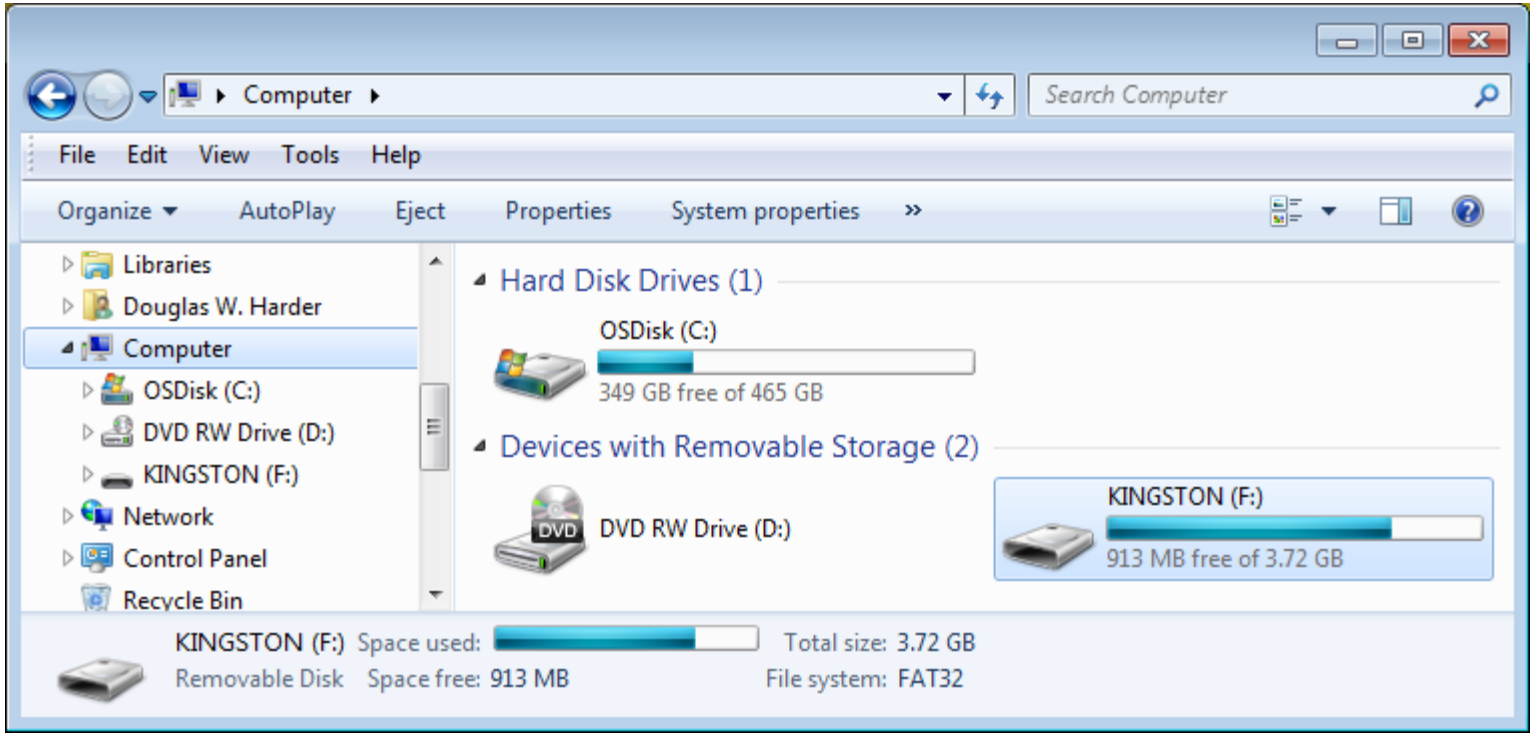
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	**1**	**2**	**3**	0	1	1	1	2	2	3	3	3	3	5	5	6	10	12	12	15	16	16	19	19	19
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z



Application

In Windows, each drive forms the root of its own directory structure

- Each of the directories is hierarchical—that is, a rooted tree



Application

In C++, if you do not use multiple inheritance, the class inheritance structure is a forest

- In Java and C#, it is a rooted tree with `Object` being the root class

If you allow multiple inheritance in C++, you have a partial order

- A *directed acyclic graph* data structure allows you store such a relation