DATA STRUCTURE

- My Introduction
- Reality of the course
- Time we will spend
- Theory + Code ratio
- My strategy
- Your devotion
- Book followed - Data structures by Seymour Lipschutz (Schaum Series)

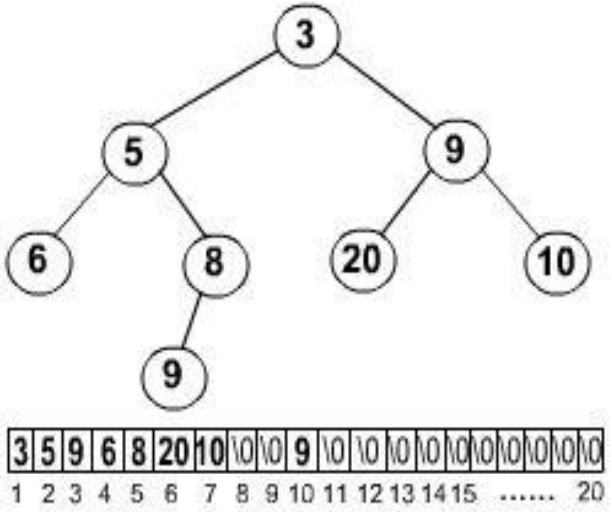
LET’S START!

Blog-
www.delvingintothelight.wordpress.com

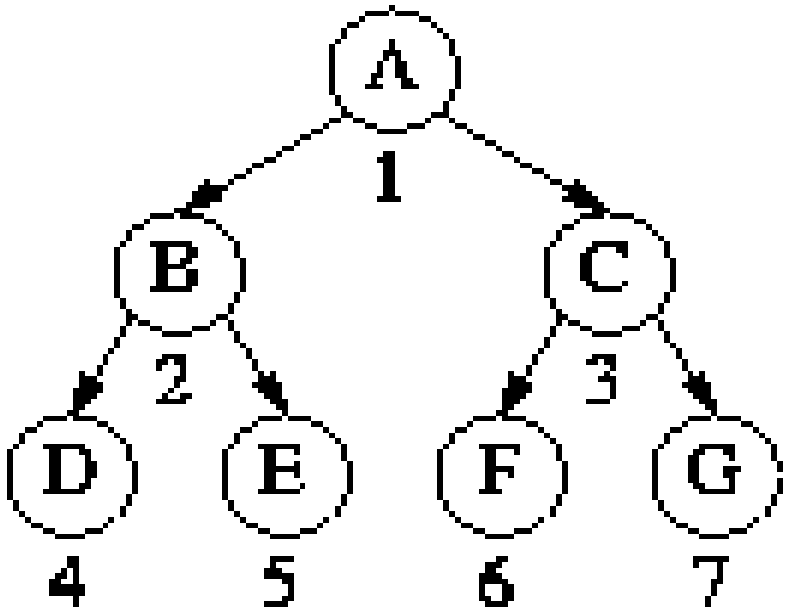
Array representation of a Tree

A single array can be used to represent a binary tree.

For these nodes are numbered / indexed according to a scheme giving 1 to root. Then all the nodes are numbered from left to right level by level from top to bottom. Empty nodes are also numbered. Then each node having an index i is put into the array as its ith element.



- Array position of root of the Tree :1
- left child of node in array position K: array position 2K
- right child of node in array position K: array position 2K+1



Drawbacks of Array Representation

1. Fixed size of Array - As we have the fixed size of the array while we don’t know the size of the tree.

2. Memory waste - A large no of memory is wasted due to unbalanced tree nodes in all the trees except the complete binary trees.

BINARY SEARCH TREE

```
#include <stdio.h>
#include <stdlib.h>

char tree[100],data;
int n=1;


void search(int i)
{
    if ((data > tree[i]) && (tree[2*i+1] != NULL))
    search(2*i+1);
    else if ((data > tree[i]) && (tree[2*i+1] == NULL))
    tree[2*i+1] = data;
    else if ((data < tree[i]) && (tree[2*i] != NULL))
    search(2*i);
    else if ((data < tree[i]) && (tree[2*i] == NULL))
    tree[2*i] = data;
}
```

```
void insert()
{

    int ch=1;
        while(ch==1){
    printf("Enter data: ");
    fflush(stdin);
        scanf("%c", &data);

    if (n == 1)
    tree[1] = data;
    else
    search(1);
    n++;
   printf("To continue press 1 else 0");
   scanf("%d",&ch);

        }
}
```

```
int inorder(int t)
{
    if (tree[t] == NULL)
        return 0;
    inorder(2*t);
    printf("%c\t", tree[t]);
    inorder(2*t+1);
}

int main()
{
        insert();
        inorder(1);

        return 0;
}
```

EXPRESSION TREE

```
#include<stdio.h>
#include<string.h>
#include<math.h>

char tree[30],s[30];

void print()
{
   int i=1,k=0,j,l,row=(strlen(tree)-1)/2;
       while(tree[i]!='\0')
       {
              for(j=1;j<=row;j++)
                     printf("  ");
               for(l=1;l<=(pow(2,k));l++)
               {
                     printf("%c  ",tree[i]);
                     i++;
              }
                     printf("\n");
                     k++;
                     row--;
              }
              i=1;
              printf("Array representation of tree\n\n");
              while(tree[i]!='\0')
              {
                   printf("%c\t",tree[i]);
                        i++;
              }

return;
}
```

```
int main(){
 int i,j=1,flag=0,sum=0;
 tree[0]='0';
 printf("Enter the Postfix Expression\n");
 gets(s);
 i=strlen(s);
 i--;
 while(i>=0)
{
        flag=0;
        if(i==(strlen(s)-1))
        {
                tree[j]=s[i];
                I--;  }
        else  {
                if(s[i]=='+' || s[i]=='-' || s[i]=='*' || s[i]=='/')
                {
                if(tree[j*2+1]=='\0')
                {
                                j=j*2+1;
                                tree[j]=s[i];
                                i--;
                }
                                else if(tree[j*2]=='\0') {
                                        j=j*2;
                                        tree[j]=s[i];
                                        i--;
                                 }      }
                        else
                        {
                        if(tree[j*2+1]=='\0')
                                {
                                        tree[j*2+1]=s[i];
                                        i--;
                                }
                        else if(tree[j*2]=='\0')
                                {
                                    tree[j*2]=s[i];
                                        i--;
                                        j=j/2;
                                }
                        } } }
print();
return 0;
}
```