University of Technology
الجامعة التكنولوجية

Computer Science Department
قسم علوم الحاسوب

Searching& Sorting Algorithms
خوارزميات البحث والترتيب

Lect. Alyaa Hasan
م.علياء حسن

cs.uotechnology.edu.iq

Searching and sorting algorithms lab

Weeks	Application title
Week1	Recursion 1
Week 2	Recursion 2
Week3	Recursion 3
Week4	Binary tree 1
Week5	Binary tree2
Week6	Binary tree3
Week7	Home works
Week8	Sequential Search algorithm
Week9	Binary Search algorithm
Week10	Quiz
Week11	Quick sort algorithm
Week12	Insertion sort algorithm
Week13	Bubble sort algorithm
Week14	Mid exam

Recursion:

الاستدعاء الذاتي : هو قابلية البرنامج الفرعي , لاستدعاء نفسه , وهو طريقة رياضية واسلوب برمجي فعال يمكن استخدامه بدل استخدام اسلوب التكرار

```
// Factorial of n = 1*2*3*...*n


int factorial (int n)
{
If (n > 1)
return n * factorial (n - 1);
else
return 1;
}
```

// The power of x^m

```
int power (int x, int m)
{
if (m == 0)
return 1;
else
return x * power(x, (m-1));
}
```

//The Fibonacci Sequence is the series of numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

في الرياضيات، متتالية فيبوناتشي أو أعداد فيبوناتشي نسبة إلى عالم الرياضيات الإيطالي ليوناردو فيبوناتشي ,،اي عدد فيها
يكون مساويا لمجموع العددين السابقين عدا العدد الاول0 = والعدد الثاني = 1

$$F_n = F_{n-1} + F_{n-2}$$

$$F_1 = 1 \text{ و } F_0 = 0$$

Example: the 8th term is the 7th term plus the 6th term:

F8 = F7 + F6

```
int fib(int n)
{
if (n <= 1)
return (n);
return fib(n-1) + fib(n-2);
}
```

//Sum of Elements Using Recursion in an array

```
#include<iostream>
using namespace std;
int sum(int a[6],int i,int &s)
{
if(i==-1)
cout<<"operation complete ";
else
{
s=s+a[i];
sum(a,i-1,s);
}
return (s);
}
int main()
{
int a[6]={47,2,56,10,13,33};
int i=5;
int s=0;
cout<<sum (a,i,s);
}
```

```
//Find First Occurrence of a Number Using Recursion in an array
#include<iostream>
using namespace std;
int first(int a[6], int x, int i){
if(i==-1){
cout<< "finsh";
}
if(a[i] == x){
return i;
}
return first(a,x,i+1);
}
int main()
{
int a[6] = {9,7,8,9,8,1};
int x = 8;
cout<<first(a,x,0);
return 0;
}
```

```
//Find  the GCD of two  numbers  , the greatest common divisor , by using recursion .

int gcd(int a, int b)
 {
   if (a == 0)
      return b;
   if (b == 0)
      return a;
   // If both numbers are equal
   if (a == b)
      return a;
   // If a is greater
   if (a > b)
     return gcd(a - b, b);

   // If b is greater
```

```
   return gcd(a, b - a); }
```

//The Maximum Element Using Recursion in an array

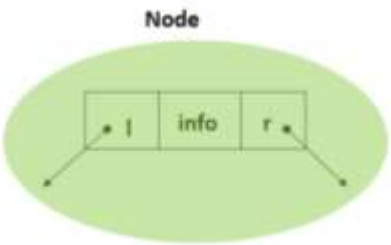
```
#include<iostream>
using namespace std;
int max(int a[6], int &x, int i) {
if(i==-1)
cout<<"finish chacking please !!";
else {
if(a[i]>x)
x=a[i];
max(a,x,i-1); }
return(x); }
int main()
{
int a[6]={12,3,45,15,62,10};
int x, i=5;
x=a[0];
max(a,x,i);
cout<<x;
}
```

Binary Search Tree

Declaration of Tree by using linked list

نفس طريقة تعريف القائمة الموصولة الفرق اضافة حقل جديد
ليصبح حقل للفرع الايمن وحقل للفرع الايسرمن نوع مؤشر.

```
struct node
{
        int info;
        struct node *l,*r;
};
typedef struct node *nodeptr;
```



Creat the BST

```
void creat(nodeptr& t)
{
  char ch;
  if(t==0)
     {
       t=new node();
       cin>>t->info;
       t->l=0;
       t->r=0;
     }
  cout<<"Do you want to add from left ("<<t->info<<") (Y,N):";
  cin>>ch;
  if(ch=='y')
    creat(t->l);
  cout<<"Do you want to add from right ("<<t->info<<") (Y,N):";
  cin>>ch;
  if(ch=='y')
    creat(t->r);
}
```

عملية خلق اول عقدة
في الشجرة (الجذر)

عملية اضافة عقدة
في الجهة اليسرى

عملية اضافة عقدة
في الجهة اليمنى

Printing the binary tree

1. In - Order Traversal

```
void inorder(nodeptr t)
{     if(t!=0)
      {
          inorder(t->l);
          cout<<t->info<<' ';
          inorder(t->r);
      }
}
```

2. Pre - Order Traversal

```
void preorder (nodeptr& t)
{     if(t!=0)
    {
       cout<<t->info<<' ';
       preorder(t->l);
       preorder(t->r);
    }
}
```

3. Post - Order Traversal

```
void postorder(nodeptr t)
{   if(t!=0)
    {
          postorder(t->l);
          postorder(t->r);
          cout<<t->info<<' ';
    }
}
```

Example/ You have a binary tree , find the leaves of the tree

```
void No(nodeptr T , int& j )
{
        if(T!=0)
        {
                if(T->L==0 && T->R==0)
                ++J;

                No(T->L, j);
                No(T->R, j);
        }
}

void main()
{
        .
        .
        .
        j=0;
        no(T,j);
        cout<<j;
}
```

Example/ You have a binary tree, find the summation of the even values in the tree.

```
void sum(nodeptr T , int& s )
{
        if(T!=0)
        {
                if(T->info%2==0)
                s=s+T->info;

                sum(T->L, s);
                sum(T->R, s);
        }
}
void main()
{
        .
        .
                s=0;
        sum(T,s);
        cout<<s;
}
```

Sorting Algorithms

1. Bubble sort الترتيب بالفقاعة

```
Void bubbleSort (int data [], int n)
{
    int temp;
  for (i = 0; i<(n-1); i++) {
    for (j = n-1; j< i; --j)
      if(data[j] < data [j-1])
      {
         temp = data [j];
         data [j] = data [j-1];
        data[j-1] = temp;
      }
}
```

2. Insertion sort algorithm الترتيب بالاختيار

```
void insertionSort(int data[], int n)
{
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = data[i];
        for  (j = i; j>0 && key <data[j-1]; j--)
              data[j ] = data[j-1];

        data[j] = key;
    }
}
```

3. Quick sort algorithm الترتيب السريع

```
Void quickSort(int data [], int first, int last)
{
    While (last > first)
    {
       int lower=first; int upper=last;
       int bound= data[first];
       while (lower< upper)
        {
           While(data[upper]> bound)
             Upper--;
          data[lower]=data[upper];
          while ((lower< upper) && (data[lower]<= bound))
              lower++;
          data[upper]= data[lower];
        }
       data[lower]= bound;
        quickSort(data, first, lower-1);
        first=lower+1;
    }
}
```

Searching Algorithms: خوارزميات البحث

1. Sequential Search algorithm خوارزمية البحث التسلسلي

```
void  sequentialsearch(int arr[size], int k)
{
    int i, pos=-1;
    for (i = 0; i < (size -1); i++)
        if (arr[i] == k)
            pos=i;

        if (pos==-1)
          cout <<"the key is not found" ;
        else
           cout <<"the key is  found in the position " << pos;

}
```

2. Binary Search algorithm خوارزمية البحث الثنائي

```
int binarySearch(int data[]), int k , int lower , int upper)
{
    int pos= -1; int mid;

    if (lower <= upper)
    {
       mid = ((lower + upper)/2);
       if k == (data[mid])
       {
          pos= mid;
          return pos;
       }

       else
       {
          if (k<data[mid])
           binarySearch(data,k,lower,mid-1);

          else
```

```
            binarySearch(data,k,mid+1,upper);

        }

    } return pos;

}
```