Iterative and recursive algorithm

An ***Iterative algorithm*** will use looping statements such as for loop, while loop or do-while loop to repeat the same steps.

An ***Iterative algorithm*** will be faster than the Recursive algorithm because of overheads like calling functions repeatedly Many times the recursive algorithms are not efficient as they take more space and time.

Factorial 5 :
5! = 5 * 4 * 3 * 2 * 1

Recursive algorithm, a module (***function***) calls itself again and again till the base condition(stopping condition) is satisfied.

Recursive algorithm is a method of simplification that divides the problem into sub-problems of the same nature. The result of one recursion is the input for the next recursion. The repletion is in the self-similar fashion. The algorithm calls itself with smaller input values and obtains the results by simply performing the operations on these smaller values.

Recursive algorithms are mostly used to solve complicated problems when their application is easy and effective.

Factorial 5 :
5! = 5 * 4! ***Where 0! =1***

Iterative and recursive Example

Try solve a problem to calculate factorial 5 using Iteration or Recursion .

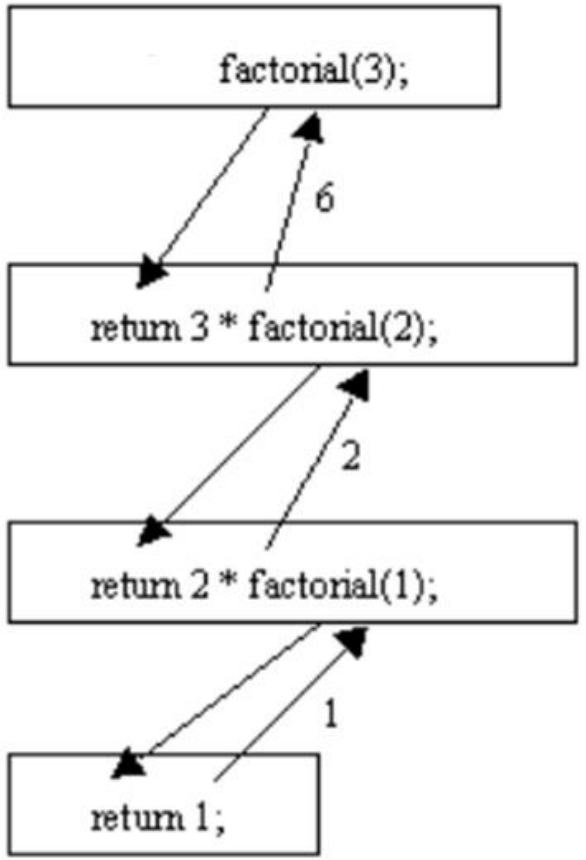
Factorial 5 :
5! = 5 * 4 * 3 * 2 * 1

```
#include <stdio.h>
void main(void)
{
    printf(" Factorial 5 = %d\n",factorial(5));
}

int factorial(int n)
{
    int f=1;
    int i ;

    for ( i=n; i>0;i--)
    {
        f = f * i;
    }
    return (f);
}
```

```
Factorial 5 = 120
```



Factorial 5 :
5! = 5 * 4! ***Where 1! =1***

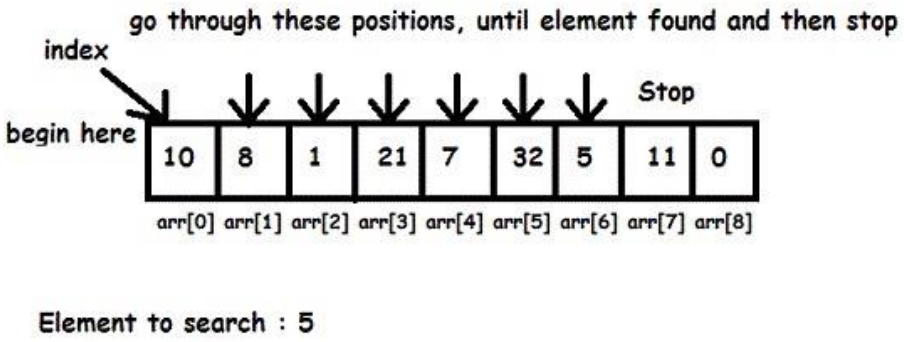
```
#include <stdio.h>
void main(void)
{
    printf(" Factorial 5 = %d\n",factorial(5));
}

int factorial(int n)
{
    if (n==1)
        return (1);
    else
        return (n*factorial(n-1));
}
```

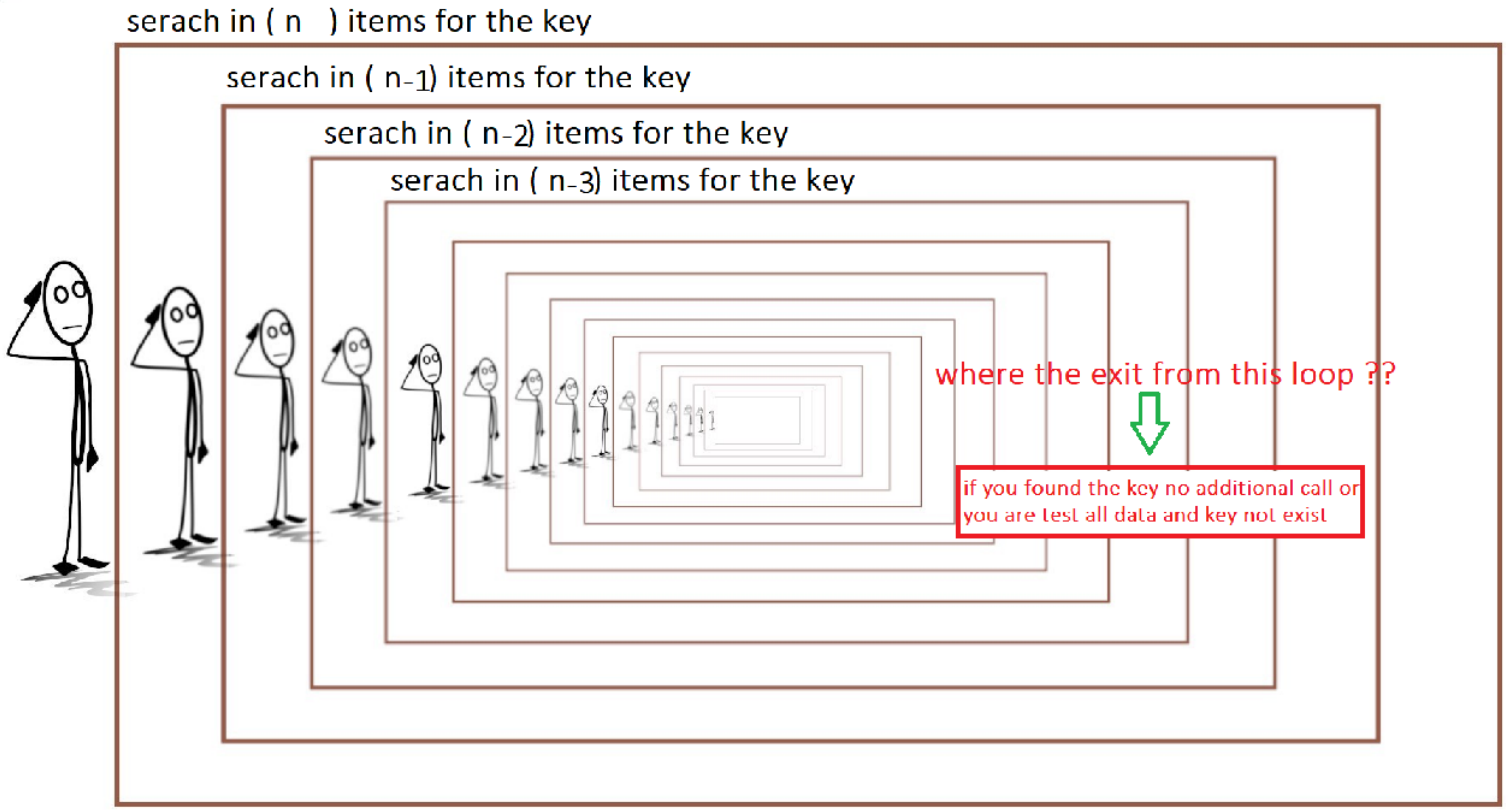
```
Factorial 5 = 120
```

Linear Search

iterative algorithm



recursive algorithm



Linear Search

```c
#include <stdio.h>
void main(void)
{
   int n=10;
   int data[10]={10,20,30,40,50,60,70,80,90,100};
   int key;
   int i;
   int location;


   printf("please value to be located:");
   scanf("%d",&key);

   location= search (data,10,key);
   if(location==-1)
      printf(" %d not located in data \n",key);
   else
      printf(" %d is located at position %d \n",key,location);

}

int search (int a[],int n,int key)
{
   int i;

   for (i=0;i<n;i++)
      if (a[i]==key) return (i);
   if (i==n) return (-1);
}
```

```c
#include <stdio.h>
void main(void)
{
   int n=10;
   int data[10]={10,20,30,40,50,60,70,80,90,100};
   int key;

   int location;


   printf("please value to be located:");
   scanf("%d",&key);

   location= search (data,10,key);
   if(location==-1)
      printf(" %d not located in data \n",key);
   else
      printf(" %d is located at position %d \n",key,location);

}

int search (int a[],int n,int key)
{
   if (n==0)
      return (-1);
   else
   {
      if (a[n] == key)
            return n;
      else
            return (search(a, n - 1, key));
   }
}
```

Binary Search

```
/* search for key in array using binary search */
pos=-1;
lower=0;
upper=n-1;

do

{
   mid = (lower + upper)/2;
   if (key==data[mid])
   {
      pos = mid;
      break;
   }
   else if (key > data[mid])
      lower=mid+1;
   else
   {
      upper=mid-1;
   }
} while (lower <= upper);

if(pos==-1)
   printf(" %f is not located in data \n",key);
else
   printf(" %f is located in position %d\n",key,pos);
```

```
   /* search for key in array using binary search */
   pos=-1;
   lower=0;
   upper=n-1;

   pos = binary_search(data,lower,upper,key);

  if (pos==-1)
      printf(" %f is not located in data \n",key);
  else
      printf(" %f is located in position %d\n",key,pos);

}

int binary_search(float a[],int start,int end,float key)
{
   int mid ;

   if (start<=end)
      mid = (start+end)/2;
   else return (-1);
   if (key ==a[mid]) return mid;
   else if (key >a[mid])
            binary_search(a,mid+1,end,key);
         else
            binary_search(a,start,mid-1,key);
}
```

Selection Sort

```c
void main(void)
{

   int data[]={19,1,200,230,12,5,222,60,50,12};
   int i;

   /* algorithm for sorting data using selection  sort  */
   print(data,10);
   sort(data,10);
   print(data,10);
}
void sort (int a[],int n)
{
   int i,j;
    int temp;

   for(i=0;i<n-1;i++)
     for (j=i+1;j<n;j++)
        if (a[i]> a[j])
              {
                 temp= a[i];
                 a[i]=a[j];
                 a[j]=temp;
              }
}
void print(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)
      printf("%d\t",a[i]);
   printf("\n");
}
```

```c
void main(void)
{

   int data[]={19,1,200,230,12,5,222,60,50,12};
   int i;

   /* algorithm for sorting data using selection  sort  */
   print(data,10);
   sort(data,0,10);
   print(data,10);
}
void sort (int a[],int start,int n)
{
   int i,j;
    int temp;

      if (start<n)
      {
         for (j=start+1;j<n;j++)
         if (a[start]> a[j])
               {
                  temp= a[start];
                  a[start]=a[j];
                  a[j]=temp;
               }
         sort(a,start+1,n);
      }
      else return ;

}
void print(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)
      printf("%d\t",a[i]);
   printf("\n");
}
```

Bubble Sort

```c
void main(void)
{
   int data[]={19,1,200,230,12,5,222,60,50,12};

   /* algorithm for sorting data using bubble sort  */
   print(data,10);
   sort(data,10);
   print(data,10);
}
void sort (int a[],int n)
{
   int sorted;
   int i;
   do
  {
     sorted=1; // true

     for (i=0;i<n-1;i++)
        if (a[i]> a[i+1])
              {
                 int temp;
                 temp= a[i];
                 a[i]=a[i+1];
                 a[i+1]=temp;
                 sorted=0; // false
              }

  } while (sorted==0);
}
void print(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)
      printf(" %d\t",a[i]);
   printf("\n");
}
```

```c
void main(void)
{
   int data[]={19,1,200,230,12,5,222,60,50,12};

   /* algorithm for sorting data using bubble sort  */
   print(data,10);
   sort(data,10);
   print(data,10);
}
void sort (int a[],int n)
{
   int sorted;
   int i;

     sorted=1; // true
     for (i=0;i<n-1;i++)
        if (a[i]> a[i+1])
              {
                 int temp;
                 temp= a[i];
                 a[i]=a[i+1];
                 a[i+1]=temp;
                 sorted=0; // false
              }
      if (sorted ==0) sort(a,n);
}
void print(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)
      printf(" %d\t",a[i]);
   printf("\n");
}
```

Merage sort

```
void main(void)
{
   int data[]={19,1,60,230,12,5,222,601,50,12};
   int temp[10];

   print(data,10);
   sort(data,temp,0,9);
   print(data,10);
}
void sort (int A[], int temp[], int low, int high )
{
   for (int k=0;k<10;k++) temp[k]=A[k];
   for (int m = 1; m <= high - low; m = 2*m)
   {
            for (int i = low; i < high; i += 2*m)
            {
               int from = i;
               int mid = i + m - 1;
               int to = min(i + 2*m - 1, high);
               merge(A, temp, from, mid, to);
            }
   }
}
void print(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)
      printf(" %d\t",a[i]);
   printf("\n");
}
void merge(int A[], int temp[], int from, int mid, int to)
{
   int k = from, i = from, j = mid + 1;

   while (i <= mid && j <= to)
   {
      if (A[i] < A[j])
         temp[k++] = A[i++];
      else
         temp[k++] = A[j++];
   }
   while (i < 10 && i <= mid)  temp[k++] = A[i++];
   for (int i = from; i <= to; i++)  A[i] = temp[i];
}
int min(int x, int y)
{
   return (x < y) ? x : y;
}
```

```
void main(void)
{
   int data[]={19,1,60,230,12,5,222,601,50,12};

   print(data,10);
   sort(data,0,9);
   print(data,10);
}
void sort (int A[], int low, int high )
{
   if(low>=high)   return;//returns recursively
   int m =low+ (high-low)/2;
   sort(A,low,m);
   sort(A,m+1,high);
   merge(A,low,m,high);
}
void print(int a[],int n)
{
   int i;
   for(i=0;i<n;i++)   printf(" %d\t",a[i]);
   printf("\n");
}
void merge(int A[], int from, int mid, int to)
{
   int temp[10];
   for (int y=0;y<10;y++) temp[y]=A[y];
   int k = from, i = from, j = mid + 1;

   while (i <= mid && j <= to)
   {
      if (A[i] < A[j]) temp[k++] = A[i++];
      else   temp[k++] = A[j++];
   }
   while (i < 10 && i <= mid)  temp[k++] = A[i++];
   for (int i = from; i <= to; i++)  A[i] = temp[i];
}
```

Sort Algorithms Comparison

```
Time elapsed is 8.642000 seconds to sort 100000 number using selection sort
Time elapsed is 27.143000 seconds to sort 100000 number using merge sort
Time elapsed is 65.263000 seconds to sort 100000 number using bubble sort
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE   10000
void m_sort (int A[], int low, int high );
void print(int a[],int n);
void main(void)
{
   int data[SIZE];
   double time_spent = 0.0;
   clock_t begin;
   clock_t end ;

   srand(time(NULL));
   fill_array(data,SIZE);
   print(data,SIZE);
   begin = clock();
   s_sort(data,SIZE);
   end = clock();
   print(data,SIZE);
   time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
   printf("Time elapsed is %f seconds to sort %d number using selection sort\n", time_spent,SIZE);

   srand(time(NULL));
   fill_array(data,SIZE);
    print(data,SIZE);
   begin = clock();
   m_sort(data,0,SIZE-1);
   end = clock();
    print(data,SIZE);
   time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
   printf("Time elapsed is %f seconds to sort %d number using merge sort\n", time_spent,SIZE);

   srand(time(NULL));
   fill_array(data,SIZE);
    print(data,SIZE);
   begin = clock();
   b_sort(data,SIZE);
   end = clock();
    print(data,SIZE);
   time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
   printf("Time elapsed is %f seconds to sort %d number using bubble sort\n", time_spent,SIZE);
}
```

selection sort

merge sort

bubble sort