4). Sort using merge sort based on id

Algorithm

Let consider some variable

List1 &lt;- first list of n employ id

List2 &lt;- second list of m employ id

List3 &lt;- merging bot sorted list

1. start

2. initialize

I&lt;- 0; j&lt;- 0; k &lt;- 0;

3. while (( I &lt; n ) and (j &lt; m) repeat thru step 4

4 . if([list1[i] &lt; list2[j]

Then list[k] &lt;- list1 [i]

I &lt;- I +1

K&lt;- k+1

Else if (List1 [i] &gt; List2 [j])

Then List[k] &lt;- List2[j]

J &lt;- j+1

K&lt;- k+1

Else[ if element of both lists are same , then insert only one of the list ]

List[k] &lt;- list1 [i]

I &lt;- I + 1

K &lt;- k+1

5. if( i&lt;n) [size of list1 is larger than list2]

6. repeat for x = I ,i+1 , … n-1

List[k] &lt;- list1 [x]

K &lt;- k + 1

7. else if(j &lt; m) [size of list2 is larger than list1]

Repeat for y = j , j+1 … m-1

List[k] &lt;- list2[j]

K &lt;- k +1

8. finished

Exit

Program:

#include&lt;stdio.h&gt;

#include&lt;conio.h&gt;

Void main()

{

Void merge\_sort(int[],int,int[],int,int[]);

int a[20] , b[20] , c[20] , n ,m ,I ,j ;

clrscr();

printf(“how many number of employ id in first list “);

scanf(“%d”,&amp;n);

printf(“\n how many number of employ id in second list “);

scanf(“%d”,&amp;m);

printf(“\n enter id of first list”);

for( I = 0; i&lt; n; I ++)

scanf(“%d”,&amp;a[i]);

printf(“\n enter id of Second list”);

for( j = 0; j&lt; m; j ++)

scanf(“%d”,&amp;b[i]);

merge\_sort(a,n,b,m,c);}

void merge\_sort(int a[] , int n , int b[] , int m , int c[])

{

Void merge\_sort(int List1 [] , int n , int List2[] , int m , int List3[])

{

Int I = j = k = 0 ,x;

While ((I &lt; n ) &amp;&amp; (j &lt; m))

{

If(List1 [i] &lt; List 2[j])

{

List3 [k] = List 1[i];

I ++;

k ++;

}

If(List1 [i] &gt; list2[j])

{

List3[k] = List2[j]

J ++;

K ++;

}

Else

{

List3[k] = List1[i]

I++;

K ++;

}

If(I &lt; n)

{

For ( int x = I ; x &lt; n ; x ++);

{

List3[k] = List1[x]

K ++;

}

}

Else if(j &lt; m)

{

For(int y = j ; y &lt; n; y ++)

{

List3[k] = List2[y];

K ++;

}

}

}

Printf(“\n display sorted list”);

For(int I = 0 ; I &lt; k ; i++)

Printf(“\n %d “ , List[k]);

}

5). Sort using Quik sort based on id

Algorithm

K <- list contain id of n record

lb <- lower bound of list

ub<- upper bound of list

key <- key value which is place in

flag <- when flag false sub list has been partition into two sub list , if flage true which indicate end of process.

Quick\_sort(k,lb,up)

1.[initialization] flage <- true

2. [perform sort]

If(lb < ub)

Then i <- lb

J <- ub +1

Key <- k[lb] (choose first element as pivot )

Repeat while (flag)

I <- I + 1

Repeat while (k[i] < key ) [ scan from right to left]

i <- i + 1

j <- j +1

repeat while (k[i] > key ) [ scan from right to left]

if( i < j)

then k [i] <-> k[j] [ interchange record]

else

flag<- false

k[lb] <-> k[j] [ interchange record]

call quick\_sort(k , lb , j-1) [sort first left sub list ]

call quick\_sort(k , j+1, ub) [ sort second right sub list)

3. finished

exit

Program:

#include<stdio.h>

#include<conio.h>

void main()

{

void Quicksort(int[],int);

int a[20],n,i;

clrscr();

printf("\n How many ID enter in list :");

scanf("%d",&n);

printf("\n enter the number in the list:");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

quick\_sort(a,n);

printf("\n display the sorted list :");

for(int i=0;i<n;i++)

printf("\n %d",a[i]);

getch();

}

void quicksort(int a[],int up)

{

int sublist(int [],int,int);

int low = 0,i;

if(up>low)

{

i = sublist(a,low,up);

quicksort(a,low,i-1);

quicksort(a,i+1,up);

}

}

int sublist(int a[],int low,int up)

{

int i,p,q,temp;

p=l+1;

q=up;

i=a[l]

while(q>=p)

{

while(a[p]=<i)

{

p++;

while(a[q]>i)

q--;

if(q>p)

{

temp = a[p];

a[p]=a[q];

a[q]=temp;

}

}

temp=a[i];

a[l]=a[q];

a[q]=temp;

return(q);

}

}