Heap_Sort

Algorithm

I void printarray (int arr[], int start, int end)

1. START
2. for (int i = start to end)
    1. print ("\t" + arr[i])
3. End for
4. STOP

II void fall (int arr[], int i)

1. START
2. ~~while (i < len)~~
    1. ~~if (2*i+1 <= len)~~
2. int lchild = 2*i , rchild = 2*l+1 ;
3. while (i < len)
    1. if (rchild <= len) //since rchild is always near index
        1. if (arr[lchild] > arr[rchild] && arr[lchild] > arr[i])
            1. i = lchild;
        2. Else if (arr[lchild] < arr[~~2*i+1~~ rchild] && arr[rchild] > arr[i])

1. i = rchild

3 End If

2. Else If (lchild <= len && arr[lchild] > arr[i])

1. i = lchild

2. End If

4. If (arr[i] > arr[i/2] && i > 1)

1. swap(arr[i], arr[i/2])

5. Else

1. break

6 End If

4. End While

5. STOP

II int heappop(int arr[])

1. START
2. int hpop = arr[1];
3. swap(1, len, arr);
4. len --
5. fall(arr, 1);
6. return hpop
7. STOP

IV void heapify (int arr[])

1. ~~int i = len/2;~~ START
2. int i = len/2
3. While (i >= 1)
    1. If (arr[i] < arr[2*i] ||
       arr[i] < arr[2*i+1])
        1. fall (arr, i)
    2. End If
    3. i--
4. End While
5. STOP

V void heappush (int arr[], int val)

1. START
2. len++
3. arr[len] = val
4. While (i > 1 && arr[i/2] < arr[i])
    1. swap (arr[i], arr[i/2])
    2. i = i/2
5. End while
6. STOP

VI

Void heapsort (int arr[])

1. START
2. create Maxheap
3. i = len (arr)
4. While (i > 1) do
    1. swap (arr[1], arr[i])
    2. i = i--
    3. j = 1
    4. While (j < i) do
        1. lchild = 2 * j
        2. rchild = 2 * j + 1
        3. if (arr[j] < arr[lchild]
            && arr[lchild] > arr[rchild])
            1. swap (arr[i], arr[lchild])
            2. j = lchild
        4. Else If (arr[j] < arr[rchild] &&
            arr[rchild] > arr[lchild])
            1. swap (arr[j], arr[rchild])
            2. j = rchild
        5. Else
            1. break
        6. End If

5. End While

5 End While

6. STOP

int main ()

1. START
2. ~~int~~ create array
3. ~~call~~ While (True) // infinite loop
    1. heapify (array);
    2. call heappush, heappop & printarray as per users choice
4. End While
5. STOP

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

int array[50] ,len, sz, start=1;


void printarray(int arr[] ,int start ,int len){
        for(int i=start ;i<=len ;i++){
                printf("%d\t",arr[i]);
        }
}

void swap(int i,int j,int arr[]){
        int tmp;
        tmp=arr[i];
        arr[i]=arr[j];
        arr[j]=tmp;
}

/*void rise(int  arr[], int i){ //maxheap recursion
        if(i>1 && arr[i]>arr[i/2]){
                swap(i , i/2 , arr);
        // printf("\t\t");
        // printarray(arr,1,len+1);
        // printf("\n");

                rise(arr , i/2);
        }
}
void fall(int arr[], int i){
        if(2*i+1<len){
                if( arr[2*i]>arr[(2*i)+1]&&arr[2*i]>arr[i] )
                        i=2*i;
                else if( arr[2*i]<arr[(2*i)+1]&&arr[(2*i)+1]>arr[i])
                        i=(2*i)+1;
        }
        else if(2*i<len&&arr[2*i]>arr[i])
                i=2*i;

    // printf("\t\t");
    // printarray(arr,1,len);
    // printf("\n");
        if( i>1 && arr[i]>arr[i/2] ){
                swap(i , i/2 , arr);
                fall(arr , i);
        }
}*/

void fall(int arr[], int i){
        while(i<len){
```

```
                if( 2*i+1<=len ){
                        if( arr[2*i]>arr[2*i+1] && arr[2*i]>arr[i] )
                                i=2*i;
                        else if( arr[2*i]<arr[2*i+1] && arr[2*i+1]>arr[i] )
                                i=2*i+1;
                }
                else if( 2*i<=len && arr[2*i]>arr[i] )
                        i=2*i;

//              printf("\t\t");
//      printarray(arr,1,len);
//      printf("\n");
                if( arr[i]>arr[i/2] && i>1)
                        swap(i ,i/2 ,arr);
                else
                        break;
        }

}

int heappop(int arr[]){
    int hpop=arr[1];
    swap(1,len,arr);
    len--;
    fall(arr,1);

    return hpop;
}

void heapify(int arr[]){
        int i=len/2;
        while(i>=1){
    //  printf("\t");
    //  printarray(arr,1,len);
    //  printf("\n");
        if(arr[i]<arr[2*i] || arr[i]<arr[2*i+1])
                    fall(arr,i);
                i--;
        }
}

void rise(int  arr[], int i){   //maxheap
        while(i>1 && arr[i/2]<arr[i]){
                swap(i ,i/2 ,arr);
                i=i/2;
        }
}

void heappush(int arr[] ,int val){
    len++;
```

```
    arr[len]=val;
    rise(arr,len);
}

void heapsort(int arr[]){
        int i = len, j, lchild, rchild;
        while(i > 1){
//              printf("\n\tsort order : ");
//         printarray(arr,1,len);
                if(arr[1] > arr[i]){
//                      printf("\n\tSwap %d <-> %d", arr[1], arr[i]);
                        swap(1,i,arr);
                }
                i--;
                j = 1;
                while(j < i){
                        lchild = 2 * j;
                        rchild = 2 * j + 1;
                        if(lchild < i){
                                if(rchild < i){
                                        if((arr[lchild] > arr[j]) && (arr[lchild] >
arr[rchild])){
                                //              printf("\n\t swap %d <-> %d",
arr[j], arr[lchild]);
                                                swap(j,lchild,arr);
                                                j = lchild;
                                        }
                                        else if((arr[rchild] > arr[j]) &&
(arr[rchild] > arr[lchild])){
                                //              printf("\n\t swap %d <-> %d",
arr[j], arr[rchild]);
                                                swap(j,rchild,arr);
                                                j = rchild;
                                        }
                                        else
                                                break;
                                }
                                else{
                                        if(arr[lchild] > arr[j]){
                                //              printf("\n\t swap %d <-> %d",
arr[j], arr[lchild]);
                                                swap(j,lchild,arr);
                                                j = lchild;
                                        }
                                        else
                                                break;
                                }
                        }
                        else
                                break;
```

```
                }
        }
//      printf("\n\n\t");
        printf("sort order: ");
    printarray(arr,1,len);
        printf("\n");
}

int main(){
        len=7;
        int array[]={-1000,5,3,8,2,7,9,1};
//      printf("enter the array size : ");
//      scanf("%d",&len);

//      printf("enter the array elements : ");
//      for(int i=1;i<=len;i++)
//          scanf("%d",&array[i]);
        heapify(array);
        sz=len;

    int choice, val;

    //printf("\n1...heapify\n");
    printf("\n0...display\n");
    printf("1...heap sort\n");
    printf("2...heap insert\n");
    printf("3...heap pop\n");
    printf("4...quit\n");

    int quit=1;
    while(quit!=0){
        heapify(array);
        printf("\nOption : ");
        scanf("%d",&choice);
        switch(choice){
            // case -1: heapify(array);
            //     printf("elements are heapsorted\n");
            //     break;
            case 0: printf("heap order: ");
                printarray(array,1,len);
                printf("\n");
                break;

            case 1:heapsort(array);
                break;

            case 2: if(len<sz){
                    printf("what do i insert ? ");
                    scanf("%d",&val);
                    heappush(array,val);
```

```
                }
                else
                    printf("heaparray is full\n");
                break;

            case 3: if(len>=1){
                    val=heappop(array);
                    printf("%d is removed\n",val);
                }
                else
                    printf("Heap empty\n");
                break;

            case 4: printf("***program terminated*** ");
                quit=0;
                break;
            default:
                // printf("\n1...heapify\n");
                            printf("\n0...display\n");
                            printf("1...heap sort\n");
                            printf("2...heapinsert\n");
                            printf("3...heapdelete\n");
                            printf("4...quit\n");
        }
    }
}
```

```
Option : 0
heap order: 9   7   8   2   3   5   1

Option : 1
sort order: 1   2   3   5   7   8   9

Option : 2
heaparray is full

Option : 3
9 is removed

Option : 3
8 is removed

Option : 3
7 is removed

Option : 1
sort order: 1   2   3   5

Option : 0
heap order: 5   2   3   1

Option : 2
what do i insert ? 5

Option : 2
what do i insert ? 9

Option : 0
heap order: 9   5   5   1   2   3

Option :
```