Signup and get free access to 100+ Tutorials and Practice Problems Start Now

Algorithms

Solve any problem to achieve a rank

View Leaderboard

Topics: Heap Sort

Heap Sort

TUTORIAL PROBLEMS

Heap Sort

Heaps can be used in sorting an array. In max-heaps, maximum element will always be at the root. Heap Sort uses this property of heap to sort the array.

Consider an array Arr which is to be sorted using Heap Sort.

- Initially build a max heap of elements in Arr.
- The root element, that is $Arr[1]$, will contain maximum element of Arr. After that, swap this element with the last element of Arr and heapify the max heap excluding the last element which is already in its correct position and then decrease the length of heap by one.
- Repeat the step 2, until all the elements are in their correct position.

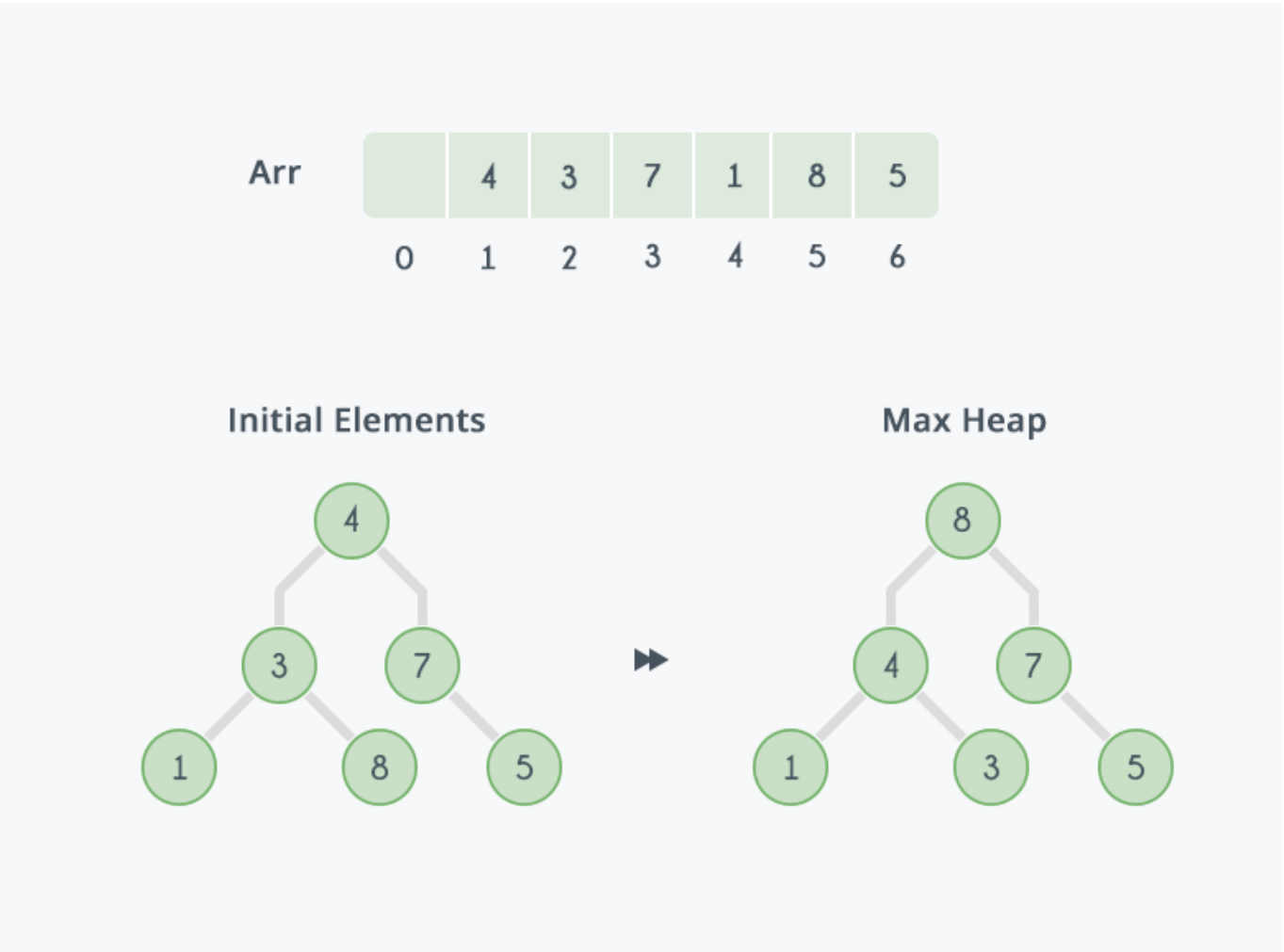
Implementation:

```
void heap_sort(int Arr[ ])

{
    int heap_size = N;

    build_maxheap(Arr);
    for(int i = N; i >= 2 ; i-- )
    {
```

```
            swap|(Arr[ 1 ], Arr[ i ]);
            heap_size = heap_size - 1;
            max_heapify(Arr, 1, heap_size);
        }
    }
```

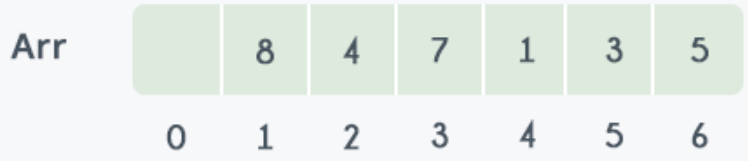
Complexity:
max_heapify has complexity $O(logN)$, build_maxheap has complexity $O(N)$ and we run max_heapify $N-1$ times in heap_sort function, therefore complexity of heap_sort function is $O(NlogN)$.

Example:
In the diagram below,initially there is an unsorted array Arr having 6 elements and then max-heap will be built.

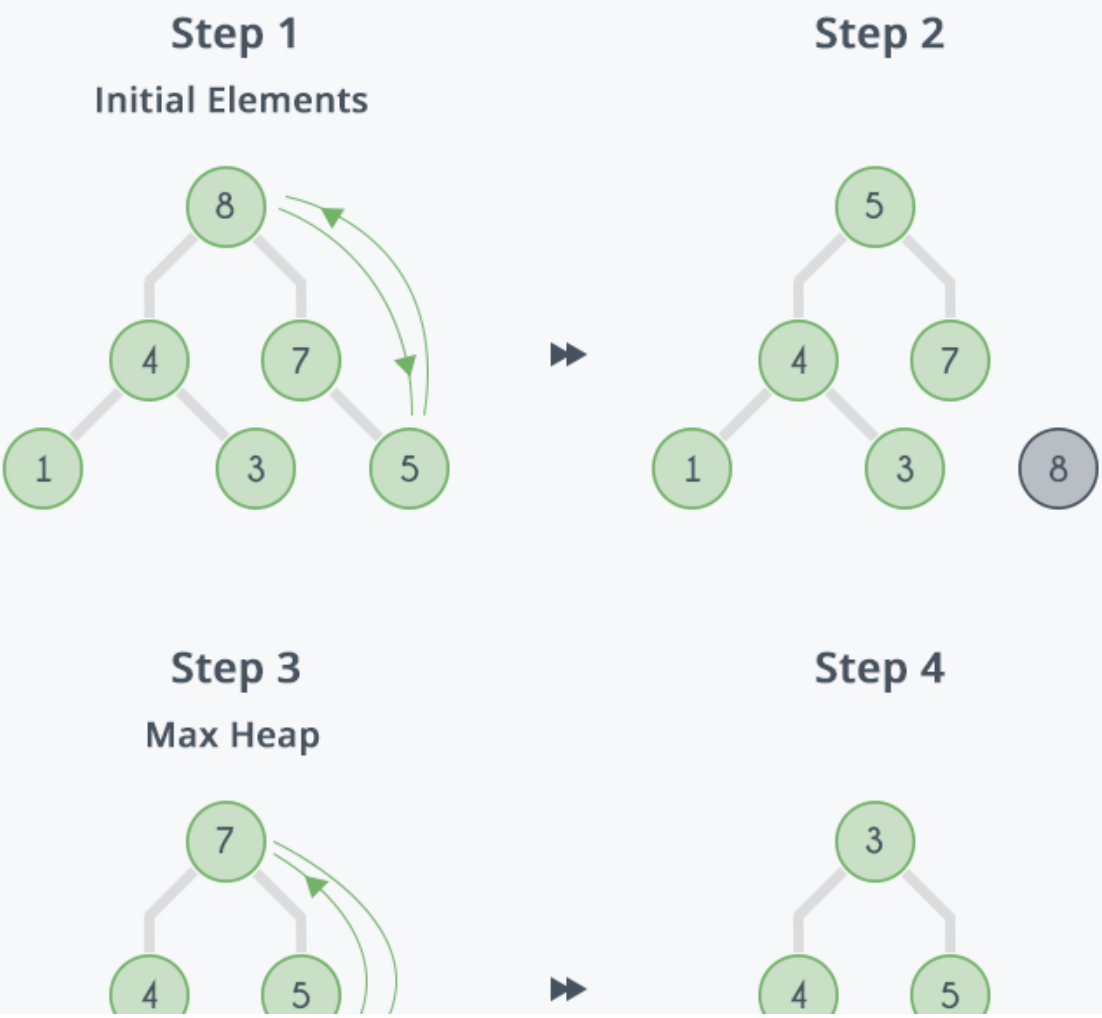


After building max-heap, the elements in the array Arr will be:

3
LIVE EVENTS
?

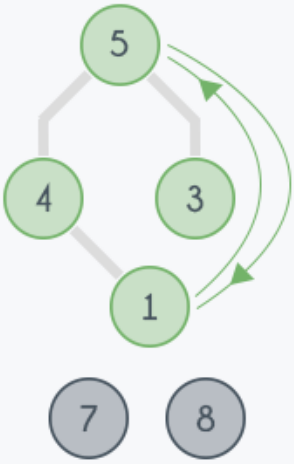


Step 1: 8 is swapped with 5.
Step 2: 8 is disconnected from heap as 8 is in correct position now and.
Step 3: Max-heap is created and 7 is swapped with 3.
Step 4: 7 is disconnected from heap.
Step 5: Max heap is created and 5 is swapped with 1.
Step 6: 5 is disconnected from heap.
Step 7: Max heap is created and 4 is swapped with 3.
Step 8: 4 is disconnected from heap.
Step 9: Max heap is created and 3 is swapped with 1.
Step 10: 3 is disconnected.

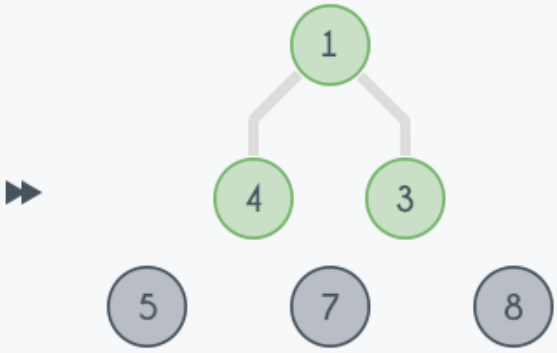


Step 5

Max Heap

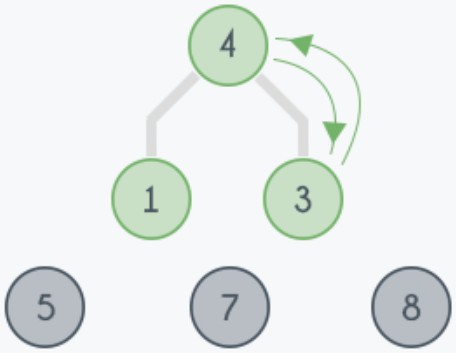


Step 6

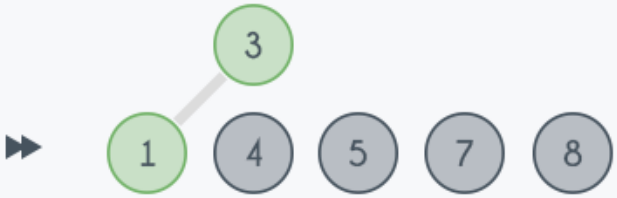


Step 7

Max Heap



Step 8

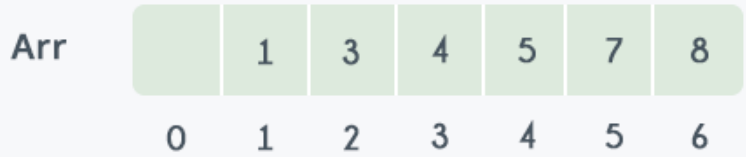


Step 9

Max Heap

Step 10

After all the steps, we will get a sorted array.



Contributed by: Akash Sharma

Did you find this tutorial helpful? YES NO

TEST YOUR UNDERSTANDING

Heaps

You are given a list of amount of money of *T* people, one by one. After each element in the list print the **top 3 richest people's amount of money**.

Input:
First line contains an integer, *T*, number of queries.
Next *T* lines contains an integer each, *X*.
i^{th} query contains amount of money i^{th} person have.

Output:
For each query, print the **top 3 richest people's amount of money** in the town and **if there are less than 3 people in town then print -1**.

Constraints:
$1 \leq T \leq 10^5$
$1 \leq X \leq 10^6$

?

SAMPLE INPUT

```
5
1
2
3
4
5
```

SAMPLE OUTPUT

```
-1
-1
3 2 1
4 3 2
5 4 3
```

Enter your code or Upload your code as file. Save C (gcc 5.4.0)

```
/*
// Sample code to perform I/O:
#include <stdio.h>

int main(){
    int num;
    scanf("%d", &num);                          // Reading input from STDIN
    printf("Input number is %d.\n", num);       // Writing output to STDOUT
}

// Warning: Printing unwanted or ill-formatted data to output will cause the test cases t
*/

// Write your code here

```

1:1

Provide custom input

COMPILE & TEST SUBMIT

COMMENTS (26) SORT BY: Relevanc

Login/Signup to Comment

Swati Lodha Edited 2 years ago

My code gives TLE. Can anyone help?

0 votes ● Reply ● Message ● Permalink

Swati Lodha 2 years ago

```
//USING MAX HEAP
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
int temp=*x;
*x = *y;
*y = temp;
}
void heapify(int a[], int n, int i)
{
int largest=i;
int l = 2*i + 1;
int r = 2*i + 2;
if(l<n && a[l]>a[largest])
largest=l;
if(r<n && a[r]>a[largest])
largest=r;
if(largest!=i)
{
swap(&a[i], &a[largest]);
heapify(a,n,largest); //recursively heapify affected sub trees
}
}
void heapSort(int a[], int n)
{
int i;
for(i=(n/2 - 1); i>=0; i--) //bottom up approach to bring max element to top
heapify(a,n,i);

for(i=n-1; i>=0; i--)
{
swap(&a[i],&a[0]); //moving current node to end
heapify(a,i,0); //heapify remaining elements
}
}
int main()
{
int i,n,count=0,j,k;
cin >>n;
int a[n];
for(i=0; i<n; i++)
{
cin >> a[i];
count++;
if(count<3)
cout << "-1\n";

else
{
heapSort(a,count);j=count-1;
```

?

```
for(k=1; k<=3; k++)
{
cout << a[j] << " ";
j--;
}
}
cout << "\n";
}


//cout << "Sorted array: ";
return 0;
}
```

0 votes ● Reply ● Message ● Permalink

Andrew Case 2 years ago

It looks like you're producing an entire sorted area with each added element. Since that involves building a max heap from the array that was sorted in the previous round, for each iteration you will be doing a O(n*lgn) operation each round for T rounds. I recommend that you preserve the max heap for each round and only get the max 3 elements from the array instead of producing the entire sorted array while preserving the max heap up until that point.

1 vote ● Reply ● Message ● Permalink

Andrew Case 2 years ago

Here's a function that I used to print out the top 3 elements of a max heap (with max element stored at index[1]).

```
void printMax3(vector <long int> &myArray) {
long int maxIndex=0,minIndex=0,max=0,left,right;
if (myArray.size() - 1 < 3)
cout << "-1" << endl;
else {

cout << myArray[1] << " ";
if (myArray[2] >= myArray[3]) {
maxIndex = 2;
minIndex = 3;
}
else {
maxIndex = 3;
minIndex = 2;
}
cout << myArray[maxIndex] << " ";
left = maxIndex * 2;
right = left + 1;
if (left < myArray.size() && myArray[left] > max) {
max = myArray[left];
maxIndex = left;
}
if (right < myArray.size() && myArray[right] >max) {
max = myArray[right];
maxIndex = right;
}
if (myArray[minIndex] > max) {
maxIndex = minIndex;
}
cout << myArray[maxIndex] << endl;
}
}
```

?

▲ 3 votes ● Reply ● Message ● Permalink

shashank kapoor a year ago

You should reuse the heapify property.
Dummy example
public void reHeapify(Integer[] arr, int size, int index) {
while (index != 0) {
if (index % 2 == 0) {
index = (index / 2) - 1;
} else {
index = index / 2;
}
heapify(arr, size, index);
}
}

▲ 0 votes ● Reply ● Message ● Permalink

Woewrut 4Ur03948039 5 months ago

#nustacoding

▲ 0 votes ● Reply ● Message ● Permalink

kashish miglani a year ago

simply use inbuilt priority_queue.
here's my code if you need any help.

```
#include<bits/stdc++.h>
#define FILL(a,b) memset((a),(b),sizeof((a)))
#define countr(v,a) (int)count(v.begin(),v.end(),a)
#define err(v) v.erase(v.begin(),v.end());
#define fast ios_base::sync_with_stdio(false),cin.tie(0),cout.tie(0);
#define in cin>>
#define ll long long
#define long_vec vector<ll>
#define nl cout<<endl;
#define out cout<<
#define print(v) repl(0,v.size()){out v[i]<<" ";}
#define rep(a,b) for(int i=a;i<b;i++)
#define repj(a,b) for(int j=a;j<b;j++)
#define repl(a,b) for(ll i=a;i<b;i++)
#define ret0 return 0;
#define sortv(v) sort(v.begin(),v.end())
#define start int main(){fast int inp;ll inpl,t,n,q,a,b,k;long_vec v;str s;
#define str string
#define pb push_back
#define vec vector<int>
#define vecp vector<pair<ll,ll>>
#define MOD 1000000007
#define MAX 1000002
using namespace std;
start
in n;
priority_queue<int> qu;
rep(0, n)
{
in inp;
qu.push(inp);
if(qu.size()<3)
{
out -1;nl
}
```

?

```
else
{
int t=3;

while(t--)
{
v.pb(qu.top());
if(t!=0)
qu.pop();
}
print(v);
rep(0, 2)
qu.push(v[i]);
v.clear();
nl
}
}
}
```

0 votes ● Reply ● Message ● Permalink

Mayank Tripathi a year ago

This works fine

```
#include <bits/stdc++.h>
using namespace std;
vector<long>heap;
int ptr=-1;
void heapify();
void swap(int,int);
int main()
{
long t,var;
cin>>t;
while(t--)
{
cin>>var;
ptr++;
heap.push_back(var);
heapify();
}
return 0;
}
void heapify()
{
int max=0,max1=0;
if(ptr<2)
{
cout<<-1<<"\n";
return;
}
int i,par;
i=ptr;
par=(i-1)/2;
while(par>=0&&heap[par]<heap[i])
{
swap(i,par);
i=par;
par=(i-1)/2;
}
max=heap[1]>heap[2]?heap[1]:heap[2];
max1=heap[1]^heap[2]^max;
```

?

```
for(i=3;i<7&&i<ptr+1;i++)
if(heap[i]>max1)
max1=heap[i];
cout<<heap[0]<<" "<<max<<" "<<max1<<"\n";
}
void swap(int i,int par)
{
int temp;
temp=heap[par];
heap[par]=heap[i];
heap[i]=temp;
}
```

0 votes ● Reply ● Message ● Permalink

Kishore Routhu a year ago

My code works

```
public static void main(String[] args) {
PrintWriter out = new PrintWriter(System.out);
InputReader in = new InputReader(System.in);
int t = in.readInt();
int a[] = new int[t];
int max1 = Integer.MIN_VALUE;
int max2 = Integer.MIN_VALUE;
int max3 = Integer.MIN_VALUE;
for (int i = 0; i < t; i++) {
a[i] = in.readInt();
if (i < 2) {
out.println(-1);
continue;
} else if (max1 == Integer.MIN_VALUE) {
heapSort(a, i + 1);
max1 = a[i];
max2 = a[i-1];
max3 = a[i-2];
} else {
if (a[i] > max1) {
max3 = max2; max2 = max1; max1 = a[i];
} else if (a[i] > max2) {
max3 = max2; max2 = a[i];
} else if (a[i] > max3) {
max3 = a[i];
}
}
out.printf("%d %d %d", max1, max2, max3);
out.println();
}
out.flush();
out.close();
}
```

It worked because the output requires TOP 3 , so that i take max1, max2, max3. By this we don't need to sort every time we take new element, sort has to be done for one time.

But how can we do for TOP 4 or TOP 5 or TOP 6 .... TOP N richest, without repeated sorting @ each new element ? can any one explain ?

2 votes ● Reply ● Message ● Permalink

vivek_23 10 months ago

This tutorial answers your question.

0 votes ● Reply ● Message ● Permalink

?

lhkzyz3 9 months ago

```
int[]top = new int[k];
for (int i = 0; i < a.length; i++) {
int tmpMax=a[i];
for (int j = 0; j < top.length; j++) {
if(tmpMax > top[j]){
int tmp =top[j];
top[j] = tmpMax;
tmpMax=tmp;
}
}
}
```

▲ 0 votes ● Reply ● Message ● Permalink

vivek_23 10 months ago

Java code => Accepted in 0.27 seconds

Tip: You may print the output for each case ALL at the end by temporarily storing it in a StringBuilder object.

Code is here - https://pastebin.com/1ukL35Kj

▲ 1 vote ● Reply ● Message ● Permalink

Sachin Joshi 5 months ago

use the inbuilt priority queue....and normal push and pop .....don't forget to reinsert the elements

▲ 1 vote ● Reply ● Message ● Permalink

Sandeep Sharma a year ago

Why for this code TLE is coming??

```
#include<iostream>
#include<vector>
#include<stdio.h>
#define ll long long
using namespace std;
ll b[3];
void heapify(ll arr[], ll n, ll i)
{
ll largest = i; // Initialize largest as root
ll l = 2*i + 1; // left = 2*i + 1
ll r = 2*i + 2; // right = 2*i + 2
// If left child is larger than root
if (l < n && arr[l] > arr[largest])
largest = l;
// If right child is larger than largest so far
if (r < n && arr[r] > arr[largest])
largest = r;
// If largest is not root
if (largest != i)
{
swap(arr[i], arr[largest]);
// Recursively heapify the affected sub-tree
heapify(arr, n, largest);
}
}
// main function to do heap sort
void heapSort(ll arr[], ll n)
{
// Build heap (rearrange array)
```

?

```
for (ll i = n / 2 - 1; i >= 0; i--)
heapify(arr, n, i);
// One by one extract an element from heap
for (ll i=n-1; i>=0; i--)
{
// Move current root to end
swap(arr[0], arr[i]);
// call max heapify on the reduced heap
heapify(arr, i, 0);
}
ll j=0;
for(ll i=n-1;i>=n-3;i--)
{
b[j]=arr[i];
j++;
}
}
int main()
{
ll n;
cin>>n;
ll a[n];
vector<ll > v[n+1];
v[1].push_back(-1);
v[0].push_back(-1);
for(ll i=0;i<n;i++)
{
cin>>a[i];
// cout<<"taken\n";
if(i>1)
{
for(ll j=0;j<3;j++)
{
b[j]=0;
}
heapSort(a,i+1);
v[i].push_back(b[0]);
v[i].push_back(b[1]);
v[i].push_back(b[2]);
}
}
for(ll i=0;i<n;i++)
{
if(i>1)
{
for(ll j=0;j<3;j++)
{
cout<<v[i][j]<<" ";
}
cout<<endl;
}
else
{
cout<<v[i][0]<<endl;
}
}
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

?

Qi Chen a year ago

```
#include <iostream>
using namespace std;
void MaxHeapFixDown(int a[], int i, int n)
{
int j, temp;
temp = a[i];
j = 2*i+1;
while(j<n)
{
if(j+1<n && a[j+1]>a[j])
j++;
if(a[j]<=temp)
break;
a[i] = a[j];
i = j;
j = 2*i+1;
}
a[i] = temp;
}
void MakeMaxHeap(int a[], int n)
{
for(int i = n/2-1; i>=0; i--)
{
MaxHeapFixDown(a, i, n);
}
}
void Swap(int &a, int &b)
{
int temp = a;
a = b;
b = temp;
}
void Top3(int a[], int *maxx)
{
maxx[0] = a[0];
maxx[1] = a[1];
maxx[2] = a[2];
if(maxx[1]<maxx[2])
Swap(maxx[1], maxx[2]);
}
void Update3(int *top3, int Value)
{
if(Value> top3[0])
{
top3[2]=top3[1];
top3[1]=top3[0];
top3[0]=Value;
}
else if(Value<top3[0] && Value>top3[1])
{
top3[2] = top3[1];
top3[1] = Value;
}
else if(Value<top3[1] && Value>top3[2])
{
top3[2] = Value;
}
```

?

```
cout << top3[0] <<" "<< top3[1]<<" "<< top3[2] << endl;
}
int main()
{
int N;
cin >> N;
int A[N];
int maxx[3];
for(int i=0; i<N; i++)
{
cin >> A[i];
if(i<2)
cout << -1 << endl;
else if(i==2)
{
MakeMaxHeap(A, 3);
Top3(A, maxx);
cout << maxx[0] <<" "<< maxx[1]<<" "<< maxx[2] << endl;
}
else
{
Update3(maxx, A[i]);
}
}
}
```

▲ 0 votes ● Reply ● Message ● Permalink

MOHD ANSARI a year ago

Can somebody help me out with my problem?
I will highly appreciate it.
Actually, the problem is that the time limit has exceeded. So, is there anyway to reduce it?

```
#include <stdio.h>
void swap(int *a,int *b){
int x=*a;
*a=*b;
*b=x;
}
void Max_heapify(int A[],int i,int heap_size){
int l,largest=i,r;
l=2*i;
r=l+1;
if(l<=heap_size&&A[l]>A[i])
largest=l;
if(r<=heap_size&&A[r]>A[largest])
largest=r;
if(largest!=i){
swap(&A[largest],&A[i]);
Max_heapify(A,largest,heap_size);
}
}
void Build_max_heap(int A[],int heap_size){
int i;
for(i=heap_size/2;i>=1;--i)
Max_heapify(A,i,heap_size);
}
int main()
{
int T,heap_size=0;
scanf("%d",&T);
int A[T+1];
```

?

```
while(T--){
++heap_size;
scanf("%d",&A[heap_size]);
Build_max_heap(A,heap_size);
if(heap_size==1||heap_size==2)
printf("%d\n",-1);
else{
printf("%d ",A[1]);
int maxindex=1,minindex=1,max=0,l,r;
if(A[2]>=A[3]){
maxindex=2;
minindex=3;
}
else{
maxindex=3;
minindex=2;
}
printf("%d ",A[maxindex]);
l=2*maxindex;
r=l+1;
if(l<=heap_size&&A[l]>max){
max=A[l];
maxindex=l;
}
if(r<=heap_size&&A[r]>max){
max=A[r];
maxindex=r;
}
if(A[minindex]>max)
maxindex=minindex;
printf("%d\n",A[maxindex]);
}
}
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Saket Agarwal 10 months ago

An easier example code:-

```
#include <iostream>
#include <algorithm>
using namespace std;
int main()
{
long long int n,temp,temp1,l1,l2,l3;
cin>>n;
long long int A[n];
for(long long int i=0;i<n;i++)
{
cin>>A[i];
if((i==0)||(i==1))
cout<<"-1"<<endl;
else if(i==2)
{
sort(A,A+i+1,greater<int>());
for(int j=0;j<3;j++)
cout<<A[j]<<" ";
cout<<endl;
l1=A[0];l2=A[1];l3=A[2];
}
```

?

```
else
{
if(A[i]>l1)
{
temp=l1;
l1=A[i];
temp1=l2;
l2=temp;
l3=temp1;
}
else if((A[i]==l1)||(A[i]>l2))
{
l3=l2;
l2=A[i];
}
else if((A[i]>l3)||(A[i]==l2))
{
l3=A[i];
}
cout<<l1<<" "<<l2<<" "<<l3<<endl;
}
}
}
```

0 votes ● Reply ● Message ● Permalink

vivek_23 10 months ago

Your code defeats the purpose of this tutorial.

0 votes ● Reply ● Message ● Permalink

karan jaju 9 months ago

```
#include <bits/stdc++.h>
using namespace std;
void heapify(int arr[],int n,int i)
{
int up=i/2;
int now=i;

if(up>=1 && arr[now]>arr[up])
{
swap(arr[now],arr[up]);
heapify(arr,n,up);
}



}
int main()
{
int t;
cin>>t;

int arr[100001]={-1000000};

cout<<"-1"<<endl;
cout<<"-1"<<endl;

cin>>arr[1];
cin>>arr[2];

for(int i=3;i<=t;i++)
```

?

```
{
int a;
cin>>a;
arr[i]=a;



heapify(arr,i,i);
int first=arr[1];

int secondind,thirdind,secondele,thirdele;
if(arr[2]>arr[3])
{secondind=2;
secondele=arr[2];}
else if(arr[2]<arr[3])
{secondind=3;
secondele=arr[3];}
else
{secondind=4;
secondele=arr[2];
thirdele=arr[3];}

if(secondind==2)
{
thirdele=max(max(arr[3],arr[4]),arr[5]);
}
else if (secondind==3)
{
thirdele=max(max(arr[2],arr[6]),arr[7]);
}

cout<<first<<" "<<secondele<<" "<<thirdele<<endl;
}


return 0;
}
```

this is the correct code with upward heapify traversal .

▲ 0 votes ● Reply ● Message ● Permalink

humblefool 9 months ago

use priority queue

▲ 0 votes ● Reply ● Message ● Permalink

Sai Pavan Nelavalli 6 months ago

C++ working code

==========

```
#include<iostream>
#include<algorithm>
#define ll long long int
using namespace std;
ll heap[2000001]={-999999999};
void insert(ll ar[],int data,int& n){
n += 1;
int lef,rig,maxi,mini;
int i = n-1;
while( i > 0 && data > ar[(i-1)/2]){
ar[i] = ar[(i-1)/2];
i = (i-1)/2;
}
```

```
ar[i] = data;
if(n<3)
cout<<"-1\n";
else
{
cout<<ar[0]<<" ";
if(ar[1] > ar[2]){
maxi = 1;
mini=2;
lef = 2*1+1;
rig = 2*1+2;
}
else{
maxi = 2;
mini=1;
lef = 2*1+1;
rig = 2*1+2;
}
cout<<ar[maxi]<<" "<<max(max(ar[lef],ar[rig]),ar[mini])<<"\n";
}
}
/*void display(ll ar[],int n){
cout<<"Updated heap : ";

for(int i = 0;i < n; i++)
cout<<ar[i]<<" ";

cout<<endl;
}*/
int main(){
int t,n,a;
cin>>t;
n = 0;
while(t--){
cin>>a;
insert(heap,a,n);
//display(heap,n);

}

return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Ravi Kumar Tahlan 6 months ago

C++ working code

==========

```
#include<iostream>
#include<algorithm>
#define ll long long int
using namespace std;
ll heap[2000001]={-999999999};
void insert(ll ar[],int data,int& n){
n += 1;
int lef,rig,maxi,mini;
int i = n-1;
while( i > 0 && data > ar[(i-1)/2]){
ar[i] = ar[(i-1)/2];
i = (i-1)/2;
}
```

?

```
ar[i] = data;
if(n<3)
cout<<"-1\n";
else
{
cout<<ar[0]<<" ";
if(ar[1] > ar[2]){
maxi = 1;
mini=2;
lef = 2*1+1;
rig = 2*1+2;
}
else{
maxi = 2;
mini=1;
lef = 2*1+1;
rig = 2*1+2;
}
cout<<ar[maxi]<<" "<<max(max(ar[lef],ar[rig]),ar[mini])<<"\n";
}
}
int main(){
int t,n,a;
cin>>t;
n = 0;
while(t--){
cin>>a;
insert(heap,a,n);
}
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Shubham Kumar 5 months ago

solution:

```
#include <bits/stdc++.h>
#include <iomanip>
using namespace std;
#define ll long long
#define ull unsigned long long
#define opt ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
#define pb push_back
#define mp make_pair
#define rep(i,n) for(i=0;i<n;i++)
#define repl(i,a,b) for(i=a;i<b;i++)
#define repa(i,a,b) for(i=a;i<=b;i++)
#define repr(i,n) for(i=n-1;i>=0;i--)
#define nl cout<<"\n";
#define M 1000000007
#define N 1000005
void solve()
{
ll t,x,a,b,c;
priority_queue<ll> pq;
cin >> t;
while(t--)
{
cin >> x;
pq.push(x);
```

```
if(pq.size()<3) cout << -1;
else
{
a=pq.top();
pq.pop();
b=pq.top();
pq.pop();
c=pq.top();
pq.pop();
pq.push(a);
pq.push(b);
pq.push(c);
cout << a << ' ' << b << ' ' << c;
}
nl;
}
}

int main()
{
opt;
solve();
return 0;
}
```

▲ 0 votes ● Reply ● Message ● Permalink

Shantanu Dwivedi 4 months ago

```
using namespace std;
#include<iostream>
#include<cmath>
#include<algorithm>
#include<vector>
int heapsize,ct;
void min_heapify(int arr[], int i)
{
int l=(2*i);
int r=(2*i+1);
int smallest;
if(l<=heapsize && arr[l]<arr[i])
{
smallest=l;
}
else
{
smallest=i;
}
if(r<=heapsize && arr[r]<arr[smallest])
{
smallest=r;
}
if(smallest==i)
return;
if(smallest!=i)
{
int swap=arr[i];
arr[i]=arr[smallest];
arr[smallest]=swap;
}
min_heapify(arr,smallest);
}
```

?

```
void build_minheap(int arr[])
{
for(int k=floor(heapsize/2);k>=1;k--)
{
min_heapify(arr,k);
}
}
void heapsort(int arr[])
{
build_minheap(arr);
for(int j=heapsize;j>=2;j--)
{
int temp=arr[1];
arr[1]=arr[j];
arr[j]=temp;
heapsize--;
ct++;
if(ct==3)
{
break;
}
min_heapify(arr,1);
}
}
int main()
{
int arr[1000000]={};
int q,flag;
cin>>q;
for(int i=1;i<=q;i++)
{
cin>>flag;
heapsize=flag;
if(flag<=2)
{
cout<<"-1"<<endl;
}
else if(flag>2)
{
for(int j=1;j<=flag;j++)
{
arr[j]=j;
}
ct=0;
heapsort(arr);
for(int k=1;k<=3;k++)
{
cout<<arr[k]<<" ";
}
cout<<endl;
}
}
}
```

My code gives TLE. Can anyone help?

▲ 0 votes ● Reply ● Message ● Permalink

Pranjal Singh 4 months ago

```
#include <iostream>
#include <algorithm>
using namespace std;
```

?

```
int* heap_sort(int* a, int x)
{
if(x % 2 == 0)
{
if(a[x] > a[x/2])
swap(a[x], a[x/2]);
x = x/2;
}

for(int i=x;i>1;i=i/2)
{
int r = i/2; int j = i-1;

if(a[i]>=a[j] && a[i]>a[r])
swap(a[i], a[r]);

if(a[j]>=a[i] && a[j]>a[r])
swap(a[j], a[r]);
}//for

return a;
}//heap_sort
void display(int* a, int x)
{
if(x == 1 || x == 2)
cout << -1 << endl;

else
{
a = heap_sort(a, x);
int max1 = a[1]; int max2,max3;
if(a[2] > a[3])
{
max2 = a[2]; max3 = a[3];
if(x>3 && a[4]>a[3])
max3 = a[4];
if(x>4 && a[4]>a[3] && a[4]>=a[5])
max3 = a[4];
if(x>4 && a[5]>a[3] && a[5]>=a[4])
max3 = a[5];
}
else
{
max2 = a[3]; max3 = a[2];
if(x>5 && a[6]>a[2])
max3 = a[6];
if(x>6 && a[6]>a[2] && a[6]>=a[7])
max3 = a[6];
if(x>6 && a[7]>a[2] && a[7]>=a[6])
max3 = a[7];
}
cout << max1 << " " << max2 << " " << max3 << endl;
}//else
}//display
int main()
{
int T;
cin >> T;

int* arr = new int[T+1];
```