**Heap Sort**

**Algorithm of Heap Sort:**

**1.** Build a max heap from the input data.  
**2.** At this point, the largest item is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of heap by 1. Finally, heapify the root of the tree.  
**3.** Repeat step 2 while size of heap is greater than 1.

**Program for Heap Sort:**

#include <iostream>

using namespace std;

void MaxHeapify (int a[], int i, int n)

{

int j, temp;

temp = a[i];

j = 2\*i;

while (j <= n)

{

if (j < n && a[j+1] > a[j])

j = j+1;

if (temp > a[j])

break;

else if (temp <= a[j])

{

a[j/2] = a[j];

j = 2\*j;

}

}

a[j/2] = temp;

return;

}

void HeapSort(int a[], int n)

{

int i, temp;

for (i = n; i >= 2; i--)

{

temp = a[i];

a[i] = a[1];

a[1] = temp;

MaxHeapify(a, 1, i - 1);

}

}

void Build\_MaxHeap(int a[], int n)

{

int i;

for(i = n/2; i >= 1; i--)

MaxHeapify(a, i, n);

}

int main()

{

int n, i;

cout<<"\nEnter the number of data element to be sorted: ";

cin>>n;

n++;

int arr[n];

for(i = 1; i < n; i++)

{

cout<<"Enter element "<<i<<": ";

cin>>arr[i];

}

Build\_MaxHeap(arr, n-1);

HeapSort(arr, n-1);

cout<<"\nSorted Data ";

for (i = 1; i < n; i++)

cout<<"->"<<arr[i];

return 0;

}