

Data Structures

Lab-12



Name: Talib Husain

Roll # 21F-9070

Task-1

#include <iostream>

using namespace std;

class Heap

{

public:

//Attrbutes

    int \*arr;

    int maxsize;

    int count;

    //constructir

    Heap(int s)

    {

        maxsize = s;

        arr = new int(s);

        count = 0;

    }

    // return left child

    int left(int i)

    {

        return (i \* 2) + 1;

    }

    // return right child

    int right(int i)

    {

        return (i \* 2) + 2;

    }

    // return parent

    int parent(int i)

    {

        return (i - 1) / 2;

    }

    // function to insert element inside heap

    void insert(int val)

    {

        if (count == maxsize)

        {

            cout << "\nOverFlow\n";

            return;

        }

        count++;

        int i = count - 1;

        arr[i] = val;

        while ((i != 0) && arr[parent(i)] > arr[i])

        {

            swap(arr[i], arr[parent(i)]);

            i = parent(i);

        }

    }

    // function to return the minimum element inside the heap

    int get\_min()

    {

        return arr[0];

    }

    // function to remove the minimum element inside the heap

    int extract\_min()

    {

        if (count <= 0)

        {

            return 0;

        }

        if (count == 1)

        {

            count--;

            return arr[0];

        }

        int root = arr[0];

        arr[0] = arr[count - 1];

        count--;

        return root;

    }

    //dis-lay

    void display()

    {

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

        {

            cout << arr[i] << "  ";

        }

    }

};

int main()

{

   int choice;

   int size,elem;

   cout<<"Enter size of Heap: ";

   cin>>size;

   Heap H(size);

       while (1){

        cout << "1-Insert\n2-Extract Min\n3-Get Min\n4-Display\n5-Exit" << endl;

        cout << "Enter Choice: ";

        cin >> choice;

        switch (choice) {

        case 1:

            cout<<"Enter Element: ";

            cin>>elem;

            H.insert(elem);

            break;

        case 2:

            cout << "Min Extracted!" << endl;

            cout<<"\nExtracted Min: "<<H.extract\_min()<<endl;         //calling function for extraction

            break;

        case 3:

            cout << "Min Value is: " << H.get\_min() << endl;    //displlays minimum value

            break;

        case 4:

            cout << "\nHeap Elements Is: ";

            H.display();     //calling display function

            break;

        case 5:

            exit(1);

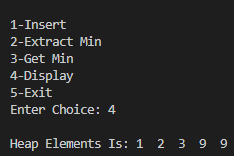
        }

    system("pause");

    system("cls");

    }

}



Text

Description automatically generated

Text

Description automatically generated

**Task-2**

#include<iostream>

using namespace std;

void heapify(int arr[], int N, int i)

{

    // largest as root

    int largest = i;

    int left = 2 \* i + 1;

    int right = 2 \* i + 2;

    // if left children is larget then the root

    if (left < N && arr[left] > arr[largest])

        largest = left;

    // If right children is larger than largest,//so far

    if (right < N && arr[right] > arr[largest])

        largest = right;

    // If the largest is not root

    if (largest != i)

    {

        swap(arr[i], arr[largest]);

        heapify(arr, N, largest);

    }

}

// for sorting

void heapSort(int arr[], int N)

{

    for (int i = N / 2 - 1; i >= 0; i--)

        heapify(arr, N, i);

    for (int i = N - 1; i > 0; i--)

    {

        swap(arr[0], arr[i]);

        heapify(arr, i, 0);

    }

}

// print array

void display(int arr[], int N)

{

    for (int i = 0; i < N; ++i)

        cout << arr[i] << " ";

    cout << "\n";

}

int main()

{

    int \*arr,size;

    cout<<"Enter array Size: ";

    cin>>size;

    arr = new int[size];

    cout<<"Enter Elements\n";

    for (int i=0;i<size;i++){

        cin>>arr[i];

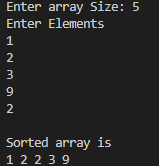
    }

    heapSort(arr, size);

    cout << "\nSorted array is \n";

    display(arr, size);

}

****

**Task-3**

#include <iostream>

using namespace std;

// to heapify a subtree with root at given index

void max\_heap(int arr[], int i, int N)

{

    int largest = i;

    int right = 2 \* i + 2;

    int left = 2 \* i + 1;

    if (left < N && arr[left] > arr[i])

        largest = left;

    if (right < N && arr[right] > arr[largest])

        largest = right;

    if (largest != i)

    {

        swap(arr[i], arr[largest]);

        max\_heap(arr, largest, N);

    }

}

// creating max heap

void build\_maxHeap(int arr[], int N)

{

    for (int i = (N - 2) / 2; i >= 0; --i)

        max\_heap(arr, i, N);

}

//displaying array

void Display(int \*Arrr, int size)

{

    for (int i = 0; i < size; ++i)

        cout << Arrr[i] << " ";

}

   int extract\_max(int \*arr,int &size)

    {

        if (size <= 0)

        {

            return 0;

        }

        if (size == 1)

        {

            size--;

            return arr[0];

        }

        int root = arr[0];

        arr[0] = arr[size - 1];

        size--;

        return root;

    }

void Kth\_Greatest(int \*arr,int n,int size){

    cout<<n<<" Max Elemennts\n";

    int max;

    for (int i=0;i<n;i++){

        max = extract\_max(arr,size);

    }

    cout<<"Kth Max: "<<max<<endl;

}

int main()

{

    int \*arr,size;

    cout<<"Enter array Size: ";

    cin>>size;

    arr = new int[size];

    cout<<"Enter Elements\n";

    for (int i=0;i<size;i++){

        cin>>arr[i];

    }

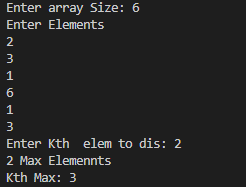
    int max;

    cout<<"Enter Kth  elem to dis: ";

    cin>>max;

    Kth\_Greatest(arr,max,size);

}

****

**Task-5**

#include <iostream>

using namespace std;

// to heapify a subtree with root at given index

void max\_heap(int arr[], int i, int N)

{

    int largest = i;

    int right = 2 \* i + 2;

    int left = 2 \* i + 1;

    if (left < N && arr[left] > arr[i])

        largest = left;

    if (right < N && arr[right] > arr[largest])

        largest = right;

    if (largest != i)

    {

        swap(arr[i], arr[largest]);

        max\_heap(arr, largest, N);

    }

}

// creating max heap

void build\_maxHeap(int arr[], int N)

{

    for (int i = (N - 2) / 2; i >= 0; --i)

        max\_heap(arr, i, N);

}

//displaying array

void Display(int \*Arrr, int size)

{

    for (int i = 0; i < size; ++i)

        cout << Arrr[i] << " ";

}

int main()

{

    int \*arr,size;

    cout<<"Enter array Size: ";

    cin>>size;

    arr = new int[size];

    cout<<"Enter Elements\n";

    for (int i=0;i<size;i++){

        cin>>arr[i];

    }

    cout << "\nMinimum Array of Heap: ";

    Display(arr, size);

    build\_maxHeap(arr, size);

    cout << "\n\nMaximum Array of Heap: ";

    Display(arr, size);

    cout<<endl;

}

**Text

Description automatically generated**