Lab a

Question 1 \_ 2

#include <iostream>

using namespace std;

class min\_heap

{

public:

int \*arr;

int capacity;

int size;

min\_heap(int c)

{

this->arr = new int[c];

this->capacity = c;

this->size = 0;

}

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

{

int smallest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[smallest] > arr[l])

{

smallest = l;

}

if (r < n && arr[smallest] > arr[r])

{

smallest = r;

}

if (smallest != i)

{

int t = arr[i];

arr[i] = arr[smallest];

arr[smallest] = t;

heapify(arr, smallest, n);

}

}

void shift\_up(int arr[], int i)

{

if (i>0)

{

if (arr[i] < arr[i / 2])

{

int t = arr[i];

arr[i] = arr[i / 2];

arr[i / 2] = t;

shift\_up(arr, i/2);

}

}

}

void insert(int d)

{

if (size == capacity)

{

cout << "overflow\n";

return;

}

arr[size] = d;

int i = size;

size++;

shift\_up(arr, i);

}

void del(int d)

{

if (size <= 0)

{

cout << "underflow\n";

return;

}

int i;

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

{

if (arr[i] == d)

{

break;

}

}

if (i == size)

{

cout << "not found data in heap\n";

return;

}

int t = arr[i];

arr[i] = arr[size - 1];

arr[size - 1] = t;

size--;

heapify(arr, i, size);

}

int extract\_min()

{

if (size <= 0)

{

cout << "underflow\n";

return -1;

}

int i = 0;

int ans = arr[i];

int t = arr[i];

arr[i] = arr[size - 1];

arr[size - 1] = t;

size--;

heapify(arr, i, size);

return ans;

}

int find\_max()

{

if (size <= 0)

{

cout << "underflow\n";

return -1;

}

int ans = 0;

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

{

if (arr[ans] < arr[i])

{

ans = i;

}

}

return arr[ans];

}

void change(int old\_value, int new\_value)

{

if (size <= 0)

{

cout << "underflow\n";

return;

}

int i;

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

{

if (arr[i] == old\_value)

{

break;

}

}

if (i == size)

{

cout << "not found data in heap\n";

return;

}

if (arr[i] > new\_value)

{

arr[i] = new\_value;

shift\_up(arr, i);

}

else

{

arr[i] = new\_value;

heapify(arr, i, size);

}

}

void print()

{

cout << "\nmin heap : ";

if (size <= 0)

{

cout << "empty";

return;

}

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

{

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

}

}

};

int main()

{

int c;

cout << "enter the capacity of min heap : ";

cin >> c;

min\_heap \*h = new min\_heap(c);

int d;

cout << "enter the value of data to insert in heap (-1 for stop) : ";

cin >> d;

while (d != -1)

{

h->insert(d);

h->print();

cout << "\nenter the value of data to insert in heap (-1 for stop) : ";

cin >> d;

}

cout << "enter the value of data to delete in heap (-1 for stop) : ";

cin >> d;

while (d != -1)

{

h->del(d);

h->print();

cout << "\nenter the value of data to delete in heap (-1 for stop) : ";

cin >> d;

}

cout << "enter the value of data for change value in heap (-1 for stop) : ";

cin >> d;

while (d != -1)

{

int k;

cout<<"enter the new value : ";

cin>>k;

h->change(d,k);

h->print();

cout << "\nenter the value of data for change value in heap (-1 for stop) : ";

cin >> d;

}

cout<<"max\_element : "<<h->find\_max();

cout<<"\nextraxt\_min : "<<h->extract\_min();

h->print();

}

Question 3

#include <iostream>

using namespace std;

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

{

int lr = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[lr] < arr[l])

{

lr = l;

}

if (r < n && arr[lr] < arr[r])

{

lr = r;

}

if (lr != i)

{

int t = arr[lr];

arr[lr] = arr[i];

arr[i] = t;

heapify(arr, lr, n);

}

}

void heap\_sort(int arr[], int n)

{

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

{

heapify(arr, i, n);

}

for (int i = n - 1; i >= 0; i--)

{

int t = arr[0];

arr[0] = arr[i];

arr[i] = t;

heapify(arr, 0, i);

}

}

int main()

{

int n;

cout << "enter the size of array : ";

cin >> n;

int arr[n];

cout << "enter value in array : ";

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

{

cin >> arr[i];

}

cout << "\ninitial array : ";

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

{

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

}

heap\_sort(arr, n);

cout << "\nsorted array : ";

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

{

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

}

}

//A stable sorting algorithm ensures that items that are equal to each other aren't reordered during the sorting process.

//it is not the stable algorithm

//Heap sort is a comparison-based sorting technique based on Binary Heap data structure.

//It is similar to the selection sort

//where we first find the minimum element and place the minimum element at the beginning.

//Repeat the same process for the remaining elements.

//Heap sort is an in-place algorithm.

Question 4

#include <iostream>

using namespace std;

class min\_heap

{

public:

int \*arr;

int capacity;

int size;

min\_heap(int c)

{

this->arr = new int[c];

this->capacity = c;

this->size = 0;

}

void shift\_up(int arr[], int i)

{

if (i>0)

{

if (arr[i] < arr[i / 2])

{

int t = arr[i];

arr[i] = arr[i / 2];

arr[i / 2] = t;

shift\_up(arr, i/2);

}

}

}

void insert(int d)

{

if (size == capacity)

{

cout << "overflow\n";

return;

}

arr[size] = d;

int i = size;

size++;

shift\_up(arr, i);

}

void print()

{

if (size <= 0)

{

cout << "empty";

return;

}

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

{

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

}

}

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

{

int smallest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[smallest] > arr[l])

{

smallest = l;

}

if (r < n && arr[smallest] > arr[r])

{

smallest = r;

}

if (smallest != i)

{

int t = arr[i];

arr[i] = arr[smallest];

arr[smallest] = t;

heapify(arr, smallest, n);

}

}

int extract\_min()

{

if (size <= 0)

{

cout << "underflow\n";

return -1;

}

int i = 0;

int ans = arr[i];

int t = arr[i];

arr[i] = arr[size - 1];

arr[size - 1] = t;

size--;

heapify(arr, i, size);

return ans;

}

};

int main()

{

int n,k;

cout<<"enter the values of k and n : ";

cin>>k>>n;

min\_heap \*h = new min\_heap(n\*k +1);

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

{

cout<<"\nenter "<<i+1<<" th sorted array : ";

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

{

int d;

cin>>d;

h->insert(d);

}

}

cout<<"merged k sorted arrays : ";

for(int i=0;i<n\*k;i++)

{

int d= h->extract\_min();

cout<<d<<" ";

}

}

//5 5

//1 2 3 4 5

//6 7 8 9 10

//60 70 80 90 100

//20 30 40 50 51

//66 67 68 69 71

Question 5

#include <iostream>

using namespace std;

class min\_heap

{

public:

int \*arr;

int capacity;

int size;

min\_heap(int c)

{

this->arr = new int[c];

this->capacity = c;

this->size = 0;

}

void shift\_up(int arr[], int i)

{

if (i > 0)

{

if (arr[i] < arr[i / 2])

{

int t = arr[i];

arr[i] = arr[i / 2];

arr[i / 2] = t;

shift\_up(arr, i / 2);

}

}

}

void insert(int d)

{

if (size == capacity)

{

cout << "overflow\n";

return;

}

arr[size] = d;

int i = size;

size++;

shift\_up(arr, i);

}

void print()

{

if (size <= 0)

{

cout << "empty";

return;

}

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

{

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

}

}

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

{

int smallest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[smallest] > arr[l])

{

smallest = l;

}

if (r < n && arr[smallest] > arr[r])

{

smallest = r;

}

if (smallest != i)

{

int t = arr[i];

arr[i] = arr[smallest];

arr[smallest] = t;

heapify(arr, smallest, n);

}

}

int extract\_min()

{

if (size <= 0)

{

cout << "underflow\n";

return -1;

}

int i = 0;

int ans = arr[i];

int t = arr[i];

arr[i] = arr[size - 1];

arr[size - 1] = t;

size--;

heapify(arr, i, size);

return ans;

}

};

int main()

{

int n;

cout << "enter the values of n : ";

cin >> n;

min\_heap \*h = new min\_heap(n + 1);

cout << "\nenter sorted array : ";

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

{

int d;

cin >> d;

h->insert(d);

}

int arr[n];

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

{

arr[i]=h->extract\_min();

}

int k;

cout<<"enter the value of k : ";

cin>>k;

if(k<n)

{

cout<<"kth largest largest element : "<<arr[n-k];

}

}

Lab b

Question 2 3 4

#include <iostream>

using namespace std;

class piority\_heap

{

public:

int \*arr;

int capacity;

int size;

piority\_heap(int c)

{

this->arr = new int[c];

this->capacity = c;

this->size = 0;

}

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

{

int smallest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[smallest] > arr[l])

{

smallest = l;

}

if (r < n && arr[smallest] > arr[r])

{

smallest = r;

}

if (smallest != i)

{

int t = arr[i];

arr[i] = arr[smallest];

arr[smallest] = t;

heapify(arr, smallest, n);

}

}

void shift\_up(int arr[], int i)

{

if (i>0)

{

if (arr[i] < arr[i / 2])

{

int t = arr[i];

arr[i] = arr[i / 2];

arr[i / 2] = t;

shift\_up(arr, i/2);

}

}

}

void insert(int d)

{

if (size == capacity)

{

cout << "overflow\n";

return;

}

arr[size] = d;

int i = size;

size++;

shift\_up(arr, i);

}

void del(int d)

{

if (size <= 0)

{

cout << "underflow\n";

return;

}

int i;

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

{

if (arr[i] == d)

{

break;

}

}

if (i == size)

{

cout << "not found data in heap\n";

return;

}

int t = arr[i];

arr[i] = arr[size - 1];

arr[size - 1] = t;

size--;

heapify(arr, i, size);

}

int extract\_min()

{

if (size <= 0)

{

cout << "underflow\n";

return -1;

}

int i = 0;

int ans = arr[i];

int t = arr[i];

arr[i] = arr[size - 1];

arr[size - 1] = t;

size--;

heapify(arr, i, size);

return ans;

}

int find\_max()

{

if (size <= 0)

{

cout << "underflow\n";

return -1;

}

int ans = 0;

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

{

if (arr[ans] < arr[i])

{

ans = i;

}

}

return arr[ans];

}

void change(int old\_value, int new\_value)

{

if (size <= 0)

{

cout << "underflow\n";

return;

}

int i;

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

{

if (arr[i] == old\_value)

{

break;

}

}

if (i == size)

{

cout << "not found data in heap\n";

return;

}

if (arr[i] > new\_value)

{

arr[i] = new\_value;

shift\_up(arr, i);

}

else

{

arr[i] = new\_value;

heapify(arr, i, size);

}

}

void print()

{

cout << "\nmin heap : ";

if (size <= 0)

{

cout << "empty";

return;

}

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

{

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

}

}

};

int main()

{

int c;

cout << "enter the capacity of min heap : ";

cin >> c;

piority\_heap \*h = new piority\_heap(c);

int d;

cout << "enter the value of data to insert in heap (-1 for stop) : ";

cin >> d;

while (d != -1)

{

h->insert(d);

h->print();

cout << "\nenter the value of data to insert in heap (-1 for stop) : ";

cin >> d;

}

cout << "enter the value of data to delete in heap (-1 for stop) : ";

cin >> d;

while (d != -1)

{

h->del(d);

h->print();

cout << "\nenter the value of data to delete in heap (-1 for stop) : ";

cin >> d;

}

cout << "enter the value of data for change value in heap (-1 for stop) : ";

cin >> d;

while (d != -1)

{

int k;

cout<<"enter the new value : ";

cin>>k;

h->change(d,k);

h->print();

cout << "\nenter the value of data for change value in heap (-1 for stop) : ";

cin >> d;

}

cout<<"max\_element : "<<h->find\_max();

cout<<"\nextraxt\_min : "<<h->extract\_min();

h->print();

}