## 实验课（十）——堆排序、top-k算法(雷镇豪)

一、实验目的

1. 了解堆的概念
2. 掌握堆的创建、插入算法以及最大堆与最小堆的调整方法
3. 掌握堆排序算法

二丶实验内容（代码）

1. Heap.h:

#include <iostream>

using namespace std;

const int max\_ = 21;//上限20个元素

struct Coordinate

{

int x;

int y;

};

bool operator > (const Coordinate &a, const Coordinate &b);

class Heap

{

public:

Heap();

virtual ~Heap();

void HeapSort();

void BuildHeap();

void Heapify(Coordinate\* arr, int i);//从i位置开始调整为最大堆

void BuildHeap\_topk();

void Heapify\_topk(Coordinate\* arr, int i, int tmpcount\_);//从第一个到第tmpcount\_个进行堆排序

int n;//坐标个数

int count\_;//当前已经进入堆的坐标个数

int k;//最小的k个数

Coordinate arr[max\_];

};

1. BinaryTree.cpp：

#include "Heap.h"

Heap::Heap()

{

//ctor

}

Heap::~Heap()

{

//dtor

}

bool operator > (const Coordinate &a, const Coordinate &b)

{

if(a.x != b.x) return a.x>b.x;

else return a.y>b.y;

}

void Heap::Heapify(Coordinate\* arr, int i)

{

int l=2\*i,r=2\*i+1,largest\_site=i;

if(l<=n && arr[l]>arr[largest\_site]) largest\_site = l;

if(r<=n && arr[r]>arr[largest\_site]) largest\_site = r;

if(largest\_site != i){

Coordinate temp = arr[i];

arr[i] = arr[largest\_site];

arr[largest\_site] = temp;

Heapify(arr,largest\_site);//从交换后的位置继续进行调整

}

}

void Heap::Heapify\_topk(Coordinate\* arr, int i, int tmpcount\_)

{

int l=2\*i,r=2\*i+1,largest\_site=i;

if(l<=tmpcount\_ && arr[l]>arr[largest\_site]) largest\_site = l;

if(r<=tmpcount\_ && arr[r]>arr[largest\_site]) largest\_site = r;

if(largest\_site != i){

Coordinate temp = arr[i];

arr[i] = arr[largest\_site];

arr[largest\_site] = temp;

Heapify\_topk(arr,largest\_site,tmpcount\_);//从交换后的位置继续进行调整

}

}

void Heap::BuildHeap\_topk()

{

cout << "请输入即将到来的坐标点个数:";

cin >> n;

cout << "请输入k:";

cin >> k;

for(count\_=1;count\_<=n;count\_++){//i是私有成员

cout << "请输入第" << count\_ << "个坐标:";

cin >> arr[count\_].x >> arr[count\_].y;

int tmpcount\_ = count\_;//保存count\_值

for(int i=count\_/2;i>=1;i--){//每次输入一个之后都先调整为最大堆

Heapify\_topk(arr,i,tmpcount\_);

}

if(tmpcount\_ >= 2){//对已经进入数组的元素进行堆排序

while(tmpcount\_ >= 2){

Coordinate temp = arr[1];

arr[1] = arr[tmpcount\_];

arr[tmpcount\_] = temp;

tmpcount\_--;

Heapify\_topk(arr,1,tmpcount\_);

}

}

if(count\_ < k) continue;//继续输入

else{

cout << "此时最小 " << k << " 项为:";

for(int i=1;i<=k;i++){

cout << "<" << arr[i].x << "," << arr[i].y << ">" << " ";

}

cout << endl;

}

}

}

void Heap::BuildHeap()

{

cout << "请输入坐标点个数:";

cin >> n;

cout << "请输入n行,每行为x与y值,并以空格分开\n";

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

cin >> arr[i].x >> arr[i].y;

}

for(int i=n/2;i>=1;i--){//从最后一个非叶子节点开始进行调整

Heapify(arr,i);

}

}

void Heap::HeapSort()

{

BuildHeap();

int tmp\_n = n;//保存最初的n

while(n>=2){

Coordinate temp = arr[1];

arr[1] = arr[n];

arr[n] = temp;

n--;//n是私有成员

Heapify(arr,1);

}

cout << "进行堆排序:";

for(int i=1;i<=tmp\_n;i++){

cout << "<" << arr[i].x << "," << arr[i].y << ">" << " ";

}

cout << endl;

}

3、main.cpp:

#include <iostream>

#include <Heap.h>

using namespace std;

int main()

{

cout << "-----------堆排序-----------" << endl;

Heap myheap1;

myheap1.HeapSort();

cout << "-----------top-k-----------" << endl;

Heap myheap2;

myheap2.BuildHeap\_topk();

return 0;

}

1. 实验结果

