Exercise 01

201918008629001 肖阳

1、

Solution:

对于复杂度为n的算法，能解决输入规模为的问题；

对于复杂度为的算法，能解决输入规模为，即的问题；

对于复杂度为的算法，能解决输入规模为，即的问题；

对于复杂度为的算法，能解决输入规模为，即的问题。

2、

Solution:

思路：

①使用桶排序对N个元素进行归纳，除去最大和最小值，剩余N-2个元素放到N-1个桶中，由抽屉原理可知必存在一个空桶，故最大间隔必定存在于两个不同的桶中。

②每个桶包含一个最大元素和最小元素，以及桶中元素的个数。第一个桶的最小元素为N个元素中最小值，第N-1个桶中最大元素为N个元素中最大值。

③将N-2个元素放到不同的桶中，同时得到每个桶中的最大元素和最小元素。

④比较相邻两个桶中后者的最小元素与前者的最大元素之差，元素个数为0的桶跳过，记录最大的差值即为最大间隔。

C++代码实现：

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

using namespace std;

class MaxInterval {

public:

int NumberOfElement;

vector<double> L;

double MaxGap;

double max, min;

struct Bucket {

double maxx = -INFINITY;

double minx = INFINITY;

int number = 0;

};

MaxInterval(const string& filename);

~MaxInterval();

double CaluculateMaxInterval(int N,vector<double> &L);

};

MaxInterval::MaxInterval(const string& filename) {

ifstream input(filename, ios::in);

char a[1000];

double cur;

if (!input) {

cout << "can't open the file" << endl;

return ;

}

input.getline(a, sizeof(a));

stringstream num(a);

num >> NumberOfElement;

input.getline(a, sizeof(a));

stringstream list(a);

for (int i = 0; i < NumberOfElement; i++) {

list >> cur;

if (i == 0) {

max = cur;

min = cur;

}

if (max < cur)

max = cur;

if (min > cur)

min = cur;

L.push\_back(cur);

}

}

MaxInterval::~MaxInterval() {

vector<double>().swap(L);//free the memory of vector

}

double MaxInterval::CaluculateMaxInterval(int N, vector<double> &L) {

double BucketSize = (max - min) / (N - 1);

vector<Bucket> B;

for (int i = 0; i < N - 1; i++) {

Bucket temp;

if (i == 0)

temp.minx = min;

if (i == N - 2)

temp.maxx = max;

B.push\_back(temp);

}

// put the element into the bucket

for (int i = 0; i < N; i++) {

int index = L[i] - min / BucketSize;

if (index >= N - 1) {

index = N - 2;

}

B[index].number++;

if (B[index].maxx < L[i]) {

B[index].maxx = L[i];

}

if (B[index].minx > L[i]) {

B[index].minx = L[i];

}

}

// find the Max Interval

if (N == 2) {

MaxGap = max - min;

}

else {

MaxGap = -INFINITY;

}

for (int i = 1; i < N - 1; i++) {

if (B[i].minx == INFINITY)

B[i].minx = B[i - 1].maxx;

if (MaxGap < (B[i].minx - B[i - 1].maxx))

MaxGap = B[i].minx - B[i - 1].maxx;

if (B[i].maxx == -INFINITY)

B[i].maxx = B[i].minx;

}

ofstream output("output.txt", ios::out);

output << MaxGap << endl;

return MaxGap;

}

int main(int argc,int \*\*argv) {

MaxInterval solution("input.txt");

double MaxGap = solution.CaluculateMaxInterval(solution.NumberOfElement, solution.L);

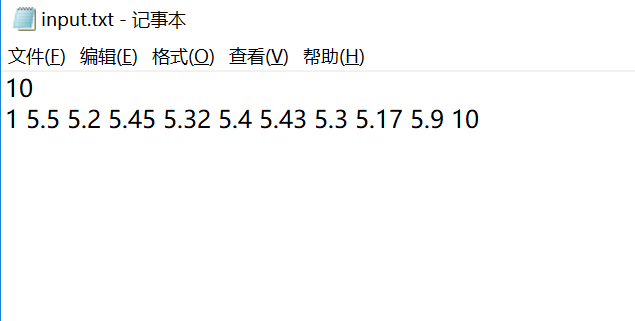
cout << MaxGap << endl;

return 0;

}

样例输入输出结果：

Input:



Output:

