第1次作业

## 第1章习题

1.5计算序列各程序段的时间复杂度。

（1）for (I=0; I<n; I++)

for (j=I; j<n;j++) x++;

（2）I=n;

While (I>1) I/= I/2;

（3）For (I=1; I<=n; I++)

For (j=1; j<=n; j++)

For (k=1; k<=n; k++)

x++;

（4）For (I=1; I<n; I++)

For (j=1; j<n; j++) x++;

For (k=1; k<n;k++) x++;

（5）For (I=1; I<n; I++)

{ j=I;

while (j<n) j\*=2;

}

（1）O(n);

（2）O(log2(n)) (以2为底n的对数)

（3）O(n^3) (n的三次方)

（4）O(n^3) (n的三次方)

（5）O(log2(n)) (以2为底n的对数)

## 第2章习题

2.7假设顺序表L中的元素按从小到大的次序排列，设计算法以删除表中重复的元素, 并要求时间尽可能少。要求：

（1）对顺序表（1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,8,9）模拟执行本算法，并统计移动元素的次数。

（2）分析算法的时间性能。

2.9递增有序顺序表A、B分别表示一个集合，设计算法求解A=A－B，并分析其时间性能。

2.7 (1)

#include<iostream>

using namespace std;

typedef struct {

int data[20];

int lengh;

}SeqList;

int main()

{

SeqList L;

L.lengh = 0;

int arr[18] = { 1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,8,9 };

for (int i = 0; i < 18; i++)

{

L.data[i] = arr[i]; L.lengh++;

}

int num = 0;

for (int i = 0; i < L.lengh - 1; )

{

if (L.data[i] == L.data[i + 1])

{

for (int j = i; j < L.lengh - 1; j++)

{

L.data[j] = L.data[j + 1];

}

num += L.lengh - i - 1; L.lengh--;

continue;

}

i++;

}

cout << "修改后顺序表内数据:";

for (int i = 0; i < L.lengh; i++)

cout << L.data[i] << " ";

cout << endl;

cout << "移动元素(覆盖)的次数为:" << num+18 << endl;

}

1. 时间复杂度O(n^2)

2.9

#include<iostream>

#define MaxSize 10

using namespace std;

typedef struct {

int data[MaxSize];

int lengh;

}SeqList;

void InitList1(SeqList& L1)//初始化

{

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

L1.data[i] = i;

L1.lengh = MaxSize;

}

void InitList2(SeqList& L2)//初始化

{

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

L2.data[i] = i + 4;

L2.lengh = MaxSize;

}

int main()

{

SeqList A, B;

InitList1(A); InitList2(B);

for (int i = 0; i < A.lengh; )

{

bool f = true;

for (int j = 0; j < B.lengh; j++)

{

if (A.data[i] == B.data[j])

{

if (i == A.lengh - 1) A.lengh--;

for (int k = i; k < A.lengh - 1; k++)

{

A.data[k] = A.data[k + 1]; A.lengh--;

}

f = false; break;

}

else if (A.data[i] < B.data[j])

break;

else;

}

if (f) i++;

}

for (int i = 0; i < A.lengh; i++)

cout << A.data[i] << " ";

cout << endl;

return 0;

}

时间复杂度O(n^3)