**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **数据结构与算法** |
| **学生姓名：** | **林墨馨** |
| **学生学号：** | **201730682485** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2018-2019学年第1学期** |

**软件学院**

**2018年9月**

# 自组织线性表以及哈希表的实现和应用

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **地 点：** | | B7 楼B座 | | 房 |  |
| **实验日期与时间：** | | 2018年11月3日 上午第1-4节 | | |
| **实验教师：** | 黄敏、齐海涛 | |

**实验内容：**

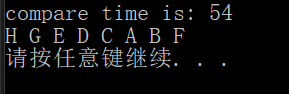
a.自组织线性表的实现和应用

本部分需要完成以下两题：

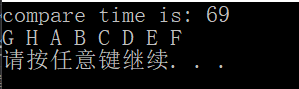
①假定值A到H存储在一个自组织线性表中，开始按照升序存放。请使用三种组织启发式规则（计数方法、移至前端方法和转置方法），按照下面顺序访问线性表，给出结果线性表和需要的比较总数。

D H H G H E G H G H E C E H G

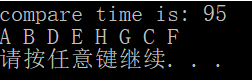
使用计数方法:



使用移至前端方法：



使用转置方法：



代码：

#include<iostream>

using namespace std;

void swap(char & a, char & b)

{

char temp;

temp = a;

a = b;

b = temp;

}

//use count way

int counter(char \* a,int size, char v,int \* time) //return compare time

{

int compare\_time = 0;//每一轮比较的次数

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

{

compare\_time++;

if (v == a[i])

{

time[i]++;

for (int j = i; j > 0; j--)

if (time[j] > time[j - 1])

{

swap(a[j], a[j - 1]);

swap(time[j], time[j - 1]);

}

return compare\_time;

}

}

return -1;

}

struct node

{

node(char d)

{

data = d;

}

node(char d, node \* n)

{

data = d;

next = n;

}

char data;

node \* next;

};

int move\_to\_front(char \* a, int a\_size, char \* v, int v\_size)//找到一条记录就将他放在最前端

{

int compare\_time = 0;

//first : make a list

node \* head = new node(a[0]);

node \* cur = head;

for (int i = 1; i < a\_size; i++)

{

cur->next = new node(a[i]);

cur = cur->next;

}

cur->next = head;

cur = head;

for (int i = 0; i < v\_size; i++)

{

compare\_time++;

while (cur->data != v[i])

{

compare\_time++;

cur = cur->next;

}

}

head = cur;

int i = 0;

a[i++] = cur->data;

//cout << cur->data << " ";

cur = cur->next;

while (cur != head)

{

a[i++] = cur->data;

//cout << cur->data << " ";

cur = cur->next;

}

return compare\_time;

}

int transpose(char \* a, int a\_size, char \* v, int v\_size)

{

int compare\_time = 0;

for (int i = 0; i < v\_size; i++)

{

for (int j = 0; j < a\_size; j++)

{

compare\_time++;

if (a[j] == v[i])

{

if (j != 1)swap(a[j], a[j - 1]);

break;

}

}

}

return compare\_time;

}

void visit(char \* c, int size)//output the final linear table

{

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

cout << c[i] << " ";

cout << endl;

}

int main()

{

char l[] = { 'A','B','C','D','E','F','G','H' };

int time[8] = {0};

char visit\_char[] = { "DHHGHEGHGHECEHG" };

int size = 8;

int visit\_time = 15;//16

int compare\_time = 0;

//使用计数方法:

//for (int i = 0; i < visit\_time; i++)

//{

//compare\_time += counter(l, size, visit\_char[i],time);

//}

//使用转置方法：

compare\_time += transpose(l, size, visit\_char, visit\_time);

//使用移至前端方法：

//compare\_time += move\_to\_front(l, size, visit\_char, visit\_time);

cout << "compare time is: " << compare\_time << "\n";

visit(l, size);

system("pause");

return 0;

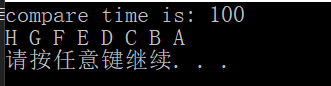
}

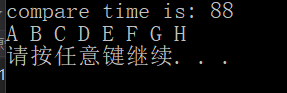
②对于三种组织启发式规则（计数方法、移至前端方法和转置方法）中的每一种，给出一组记录访问顺序，使得按照这个规则需要的比较数最多。

ABCDEFGH

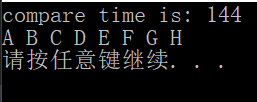
计数方法:

H G F E D C B A H G F E D C B A (每个元素访问的次数一样，每次得访问到最后一个元素才命中)

使用计数方式的运行截图

移至前端方法 HGHGFEFEHGHG（线性顺序被重复多次）

转置方法：HGHGHGHGHGHGHGHGHGHG(访问顺序不停的在H G 之间交替)



b.哈希表的实现和应用

本部分需要完成两题，第一题和第二题均是任选一题实现。

第一题，从下面两题中，任选一题实现：

①已知某哈希表的装载因子小于1，哈希函数H（key）为关键字(标识符)的第一个字母在字母表中的序号，处理冲突的方法为线性探测开放定址法。试编写一个按第一个字母的顺序输出哈希表中所有关键字的算法。

②假设哈希表长为m，哈希函数为H(x)，用链地址法处理冲突。试编写输入一组关键字并构造哈希表的算法。

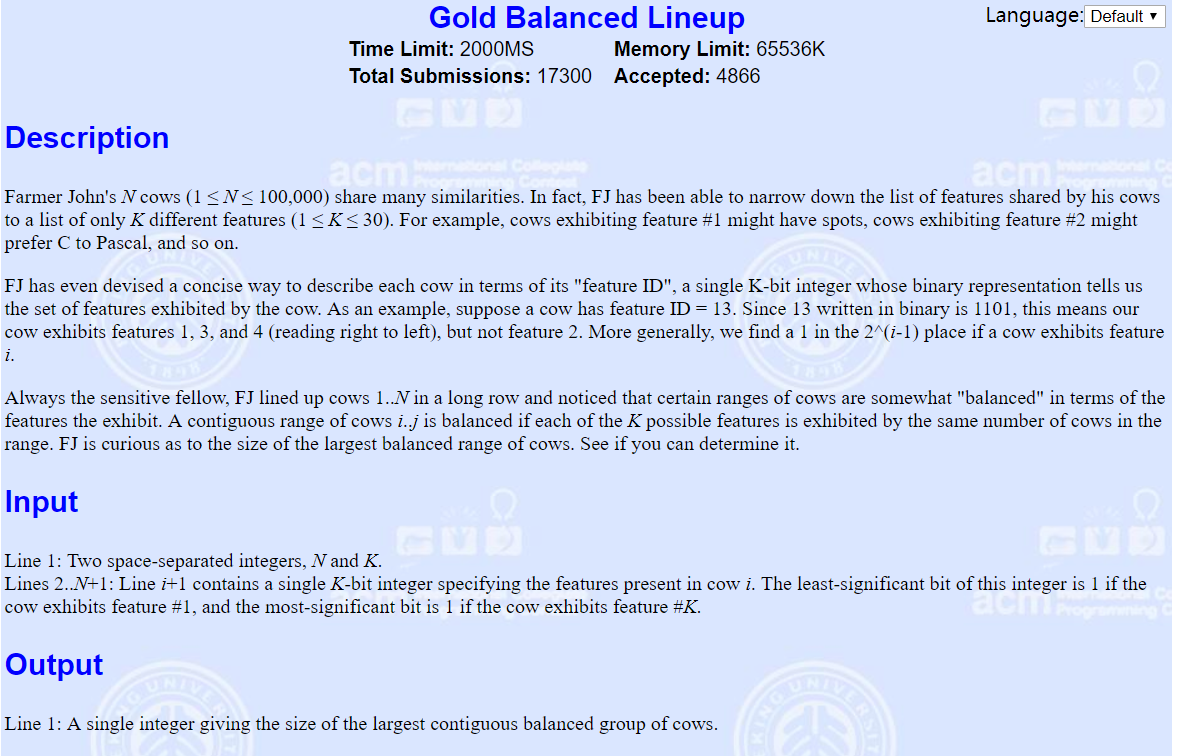
第二题，从下面两题中，任选一题实现：

①在字符串中找出第一个只出现一次的字符。如输入“abaccdeff”，则输出‘b’。

②数组中有一个数出现的次数超过了数组长度的一半，请找出这个数。

**c.附加题**

本部分需要完成下面一道题，题目链接：<http://poj.org/problem?id=3274>



**（电子版实验报告内容打印页）**