2016级数据结构第二次上机解题报告（助教版）

A:

这一题考察的是固定大小的顺序表的基本操作，具体实现书上都有。

参考代码：

#include <iostream>

#include <cstdio>

#include <fstream>

#define in std::cin

#define out std::cout

const int SIZE = 500;

struct SqList {

int data[SIZE];

int length;

int sum;

SqList() {

length = 0;

sum = 0;

}

void init() {

length = 0;

sum = 0;

}

bool ins(int x, int pos) {

if (length < SIZE and pos >= 0 and pos <= length) {

for (int i = length - 1; i >= pos; --i) {

data[i + 1] = data[i];

}

data[pos] = x;

length += 1;

sum += x;

return true;

}

return false;

}

bool del(int x) {

int index = find(x);

if (index != length) {

for (int i = index; i < length; ++i) {

data[i] = data[i + 1];

}

sum -= x;

length -= 1;

return true;

}

return false;

}

int find(int x) {

int index = 0;

for (index = 0; index < length; ++index) {

if (data[index] == x) {

break;

}

}

return index;

}

};

int main() {

SqList list;

int n;

char cmd[4];

while (in >> n) {

list.init();

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

in >> cmd;

if (cmd[0] == 'i') {

int x;

int pos;

in >> x >> pos;

if (!list.ins(x, pos)) {

out << "invalid operation!\n";

}

} else if (cmd[0] == 'd') {

int x;

in >> x;

if (!list.del(x)) {

out << "invalid operation!\n";

}

} else {

if (!list.length) {

out << "invalid operation!\n";

} else {

out << list.sum << "\n";

}

}

}

}

return 0;

}

B

题目的要求是把链表里的元素去重，不过是一个已排好序的链表，所以只需在建表后，遍历一遍，把重复的元素输出即可，不过记得在程序结束后要删除链表。

参考代码：

#include <iostream>

#include <fstream>

#define in std::cin

#define out std::cout

struct SqList {

int val;

SqList \*next;

SqList(): val(-1), next(nullptr){}

SqList(int val): val(val), next(nullptr){}

};

void show(SqList \*head) {

while (head) {

out << head->val << " ";

head = head->next;

}

out << "\n";

}

void del(SqList \*head) {

while (head) {

SqList\* temp = head;

head = head->next;

delete (temp);

}

}

int main() {

int n;

while (in >> n) {

SqList \*list, \*tail;

list = new SqList();

tail = list;

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

int val;

in >> val;

SqList \*temp = new SqList(val);

tail->next = temp;

tail = temp;

}

show(list->next);

SqList \*pre = list;

SqList \*now = pre->next;

while (now->next) {

if (now->val == now->next->val) {

pre->next = now->next;

} else {

pre = pre->next;

}

now = now->next;

}

show(list->next);

del(list);

}

return 0;

}

C：

字符串处理，可使用链表或数组等方式。以字符串（单词）为单位，遍历，当遇到当前总长度大于给定的限制时说明需要换行了，对在这行的字符串计算总长度并算出空格数量。最后对最后一行再按题目要求处理即可。

代码：

#include <iostream>

#include <string>

#include <vector>

using namespace std;

vector<string> s;

int main()

{

int n,cnt,add,tt,ttt;

string t;

cin>>n;

while(cin>>t)

s.push\_back(t);

cnt=0;

add=0;

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

{

if(add+s[i].length()+cnt<=n)

{

add+=s[i].length();

cnt++;

}

else

{

if(cnt==1)

cout<<s[i-1]<<endl;

else

{

for(int j=i-cnt;j<i-1;j++)

{

cout<<s[j];

for(int k=0;k<(n-add)/(cnt-1);k++)

cout<<" ";

}

for(int k=0;k<(n-add)%(cnt-1);k++)

cout<<" ";

cout<<s[i-1]<<endl;

}

cnt=1;

add=s[i].length();

}

}

for(int i=s.size()-cnt;i<s.size()-1;i++)

cout<<s[i]<<" ";

cout<<s[s.size()-1]<<endl;

return 0;

}

D：

双向链表，这样可以迅速查到要操作的应用程序的位置，以及进行交换。当然用数组模拟双向链表或者就是搞一个数组记录每个应用程序之前的应用程序也是可以的。时间复杂度O(m)

代码：

#include <cstdio>

long long pre[100002],pos[100002],next[100002],a[100002];

int main()

{

long long n,m,k,b,ans,p1,p2,p3,p4;

while(scanf("%lld%lld%lld",&n,&m,&k)!=EOF)

{

for(long long i=1;i<=n;i++)

scanf("%lld",&a[i]);

next[0]=a[1];

for(long long i=1;i<=n;i++)

{

pre[a[i]]=a[i-1];

pos[a[i]]=i;

next[a[i]]=a[i+1];

}

next[a[n]]=n+1;

pre[n+1]=a[n];

ans=0;

for(long long i=0;i<m;i++)

{

scanf("%lld",&b);

ans+=(pos[b]-1)/k+1;

if(pos[b]!=1)

{

p1=pre[pre[b]];

p2=pre[b];

p3=b;

p4=next[b];

pos[p3]--;

pos[p2]++;

pre[p3]=p1;

next[p3]=p2;

pre[p2]=p3;

next[p2]=p4;

next[p1]=p3;

pre[p4]=p2;

}

}

printf("%lld\n",ans);

}

return 0;

}

E

数学，递推思想。

出发点：构造结果ans[n]=num[1]+……+num[6];其中num[i]表示以i为最后一个房间的容量时所有构造的情况；计算出num[i]，从而得出结果。

首先进行初始化分析，即n=3时。

“相邻的两个房间的人数之差不能大于4”——1，6这两个房间不能相邻；而2，3，4，5这四个房间等价，且之后n=4，5，……100000的计算过程中均有此规律。即num[1]=num[6]，num[2]=num[3]=num[4]=num[5]，在写代码时注意到这点便可以不需要用到数组；

num[1]=num[6]=16;num[2,3,4,5]=18;

ans[3]=104;

再由n-1递推分析n的情况：

1、 当前面n-1个排列是房间的排列，则

A、 对2，3，4，5作为第n个教室的容量来说都能满足题意，有num[2,3,4,5]=ans[n-1];

B、 对1，6（1，6等价，记号不同而已）来说，第n-1个教室的容量不能为6，1，即要去掉几个不符合题意的组合；num[1]=ans[n-1]-num\_\_[6](前n-1个中最后一个为6的个数，)。同理 num[6]=ans[n-1]-num\_\_[1](……)。也即num[1,6]=ans[n-1]-num\_\_[6](……);

2、 当前面n-1个排列不是钥匙的排列，则

A、对i（i=2，3，4，5）作为第n个房间的容量来说能满足自习室的要求，则说明前面n-1个排列里仅有两类房间容量，且与i不同，加上i就刚好3类满足题意。那么前面两类房间容量的选法总数是从其余5类里选出两类，即C（5，2），但1，6不能同时选，故组合数为C（5，2）-1。 再看排列数，n-1个位置，每个位置可任选两类，但不能全部是同一类高度，故排列数2^(n-1)-2。

B、对i=1，6，同上面分析。因为1，6等价，所以我这里举i=1来说，前面两类房间容量里我有两种取法，选6和不选6。

对于选6，组合数是C（4，1）（剩下2，3，4，5任意选一）；再看排列数，每个位置可任选两类，但不能全部是同一容量，且最后一个也即第n-1个位置处不能为6，也可换个说法，最后一个位置放i（i=2，3，4，5），前面n-2个位置任选6和i放，但不能全放i，排列数2^(n-2)-1。

对于不选6，组合数是C（4，2）；再看排列数，每个位置可任选两类，且不能全部是同一类容量，排列数2^(n-1)-2。

把上面的组合数与排列数相乘便得到一种情况下的num[i]的值，所有情况的值相加便得到结果。详情请看代码：

#include <cstdio>

int a[100001];

int main()

{

int t16=16,t2345=18,ans=104,a\_2=4;

a[3]=ans;

for(int i=4;i<=100000;++i)

{

t16=(ans-t16+4\*(a\_2-1)+6\*(a\_2\*2-2))%7777777;

t2345=(ans+(a\_2\*2-2)\*9)%7777777;

a\_2\*=2;

a\_2%=7777777;

ans=t16\*2+t2345\*4;

ans%=7777777;

a[i]=ans;

}

int n,t;

scanf("%d",&t);

while(t--)

{

scanf("%d",&n);

printf("%d\n",(a[n]+7777777)%7777777);

}

return 0;

}

F

十字链表，如果把矩阵用向右和向下的指针串起来会发现对于每一次的交换，两个子矩阵的内部指针指向都是不变的，只需要更改四周一圈的指针指向就可以了。复杂度O((n+m)\*q)。

代码：

#include <cstdio>

#include <algorithm>

using namespace std;

typedef struct node

{

int num;

node\* down;

node\* right;

}node;

node mat[1002][1002];

int main()

{

int n,m,q,a,b,c,d,h,w;

node \*p1,\*p2,\*t1,\*t2;

while(scanf("%d%d%d",&n,&m,&q)!=EOF)

{

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

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

{

mat[i][j].down=&mat[i+1][j];

mat[i][j].right=&mat[i][j+1];

}

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

for(int j=1;j<=m;j++)

scanf("%d",&mat[i][j].num);

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

{

scanf("%d%d%d%d%d%d",&a,&b,&c,&d,&h,&w);

p1=&mat[0][0];

p2=&mat[0][0];

for(int j=0;j<a-1;j++)

p1=p1->down;

for(int j=0;j<b-1;j++)

p1=p1->right;

for(int j=0;j<c-1;j++)

p2=p2->down;

for(int j=0;j<d-1;j++)

p2=p2->right;

t1=p1;

t2=p2;

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

{

t1=t1->right;

t2=t2->right;

swap(t1->down,t2->down);

}

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

{

t1=t1->down;

t2=t2->down;

swap(t1->right,t2->right);

}

t1=p1;

t2=p2;

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

{

t1=t1->down;

t2=t2->down;

swap(t1->right,t2->right);

}

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

{

t1=t1->right;

t2=t2->right;

swap(t1->down,t2->down);

}

}

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

{

t1=&mat[i][0];

for(int j=0;j<m-1;j++)

{

t1=t1->right;

printf("%d ",t1->num);

}

printf("%d\n",t1->right->num);

}

}

return 0;

}