**2016级算法第一次上机解题报告**

16211093 林未

**一、总结**

本次上机主要考察知识点为**分治、递推、简单数论**等算法知识。

这一次上机题目助教出的很友好，难度适中，在当前学习进度下考查知识点基础且全面。但是由于错过了最后延长的30分钟，我没有能够AK十分遗憾。不过在上机结束后，我认真总结了上机时的经验教训，对剩余的题目也有了更加清晰的思考，最终全部AC。在这次报告里，我会将每一道题的思路和做题时的感受分享给大家，以供参考与交流。

**二、解题报告**

**A The stupid owls**

**思路和算法分析**

本题是错位排序问题。

首先我们来讲一讲什么是“装错信封问题”。这是由著名数学家约翰·伯努利的儿子丹尼尔·伯努利提出来的，大意如下：一个人写了n封不同的信及相应的n个不同的信封，他把这n封信都装错了信封，问都装错信封的装法有多少种。瑞士数学家欧拉称这样的问题为“全借位排列”，并给出了递推公式：f(n) = (n-1){f(n-1)+f(n-2)}。下面我来解释一下这个公式是如何得到的。

假设n个信封分别为A、B、C……，n封信分别为a、b、c……，将n封信全部装错信封的种数为f(n)。那么它包含了两种装错信的情况：**（1）**a装入B中，b装入A中，其余的信与信封都装错，此时错装种数为f(n-2)；**（2）**a装入B中，b装入除A外的信封中，此时问题变成将信b、c、d……分别错位装入信封A、C、D……中，错装种数为f(n-1)。将两种情况合并，同时又存在a装入C中 、a装入D中等n-1种情况，故总错装数f(n) = (n-1){f(n-1)+f(n-2)}。[1]

在这道题中，求的是错排的概率。又装信的方法总数为n!，因此概率公式为(f(n)/n!)\*100%。在本题中，f(n)在init()函数里求出，而n!则直接在main函数中通过for循环相除，即只需利用递推求出f(n)和n!，便可得到答案，时间复杂度为O(n)。

### 参考代码

/\*

Author: 林未

Result: AC Submission\_id: 317402

Created at: Fri Oct 13 2017 19:19:12 GMT+0800 (CST)

Problem\_id: 871 Time: 1 Memory: 1456

\*/

#include<cstdio>

#include<cstring>

using namespace std;

double f[28];

void init()

{

int i;

f[1]=0;

f[2]=1;

for(i=3;i<=26;i++) //错排

f[i]=(i-1)\*(f[i-1]+f[i-2]);

}

int main()

{

int n;

double ans;

init();

while(~scanf("%d", &n))

{

ans=f[n];

for(int i=n;i>=1;i--)

{

ans/=i;

}

ans\*=100;

printf("%.2lf%\n", ans);

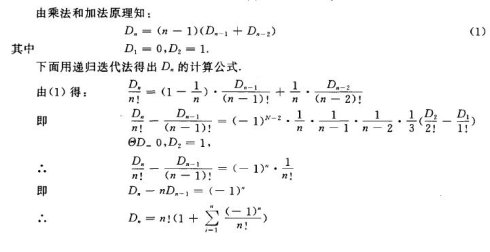
}

return 0;

}

### 注

事实上，可以利用递归迭代法求出f(n)：（图中用Dn表示f(n)）



因此，错排概率为Dn/n! = [(-1)^2/2! + … + (-1)^(n-1)/(n-1)! + (-1)^n/n!].\*100%。直接利用for循环求出该公式的解即可。

### B ModricWang和数论

**思路和算法分析**

本题是一道简单的数论题。

一开始做这道题的时候，我尝试着手写出了a分别取1~10时的答案，并顺利找到了规律：ans = (a+1)/2+1，时间复杂度为O(1)。

在上机结束后，我抱着好奇的态度将这个问题重新审视了一遍，后来终于弄懂了这个公式是怎么得到的。**首先**，假设一个整数为a，那么a%1的值一定为0，a%a的值一定为0，a%(a+1)的值一定为a，这时出现了2个不同的值0、a；**其次**，除第一点外，a%b的值不会超过a/2-1（若a为奇数则不会超过(a-1)/2-1），而不难发现从1到a/2-1（或(a-1)/2-1）中的每个数都有可能取到。因此这时出现了(a-1)/2-1+1个不同的值（根据C++整数除法规则可知a为偶数时该公式也成立）。综上，总个数ans = 2+(a-1)/2-1+1 = (a+1)/2+1 = (a+3)/2，与上机时总结的规律相一致。

### 参考代码

/\*

Author: 林未

Result: AC Submission\_id: 316819

Created at: Fri Oct 13 2017 18:37:04 GMT+0800 (CST)

Problem\_id: 843 Time: 3 Memory: 2684

\*/

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

int main()

{

long long a;

cin>>a;

if(a==1 || a==2) cout<<2<<endl;

else

{

cout<<(a+1)/2 + 1<<endl;

}

return 0;

}

### C AlvinZH去图书馆

**思路和算法分析**

本题是一道非常典型的递推题。

这道题乍一看像是斐波拉契数列的变形，殊不知其中暗藏杀机。主人公除了常规的不跨石砖或跨一个石砖外，竟然可以直接跨过两个石砖，到达第四块石砖，但是不能连续两次这种操作。用数学递推式表示，设f(n)为从第一块石砖出发走过n块石砖的小路的安全走法总数，那么他可以**（1）**不跨石砖，相当于从第二块砖出发，方法数为f(n-1)；**（2）**跨一个石砖，相当于从第三块砖出发，方法数为f(n-2)；**（3）**跨两个石砖，但此时不相当于从第四块砖出发，这种情况也直接增加了本题的难度。

在上机时我的考虑是：既然已经跨了两块石砖，则说明下一次不能再跨两块石砖，那么对于第三种情况，方法数为f(n-3)-f(n-6)，即只需从f(n-3)中方法中减去再跨两块石砖到达第六块砖的方法数即可。然而事实证明这种算法是错误的。因为这个算法将某些情况忽略了。f(n-6)已经将一开始跨两块砖的情况减去了，而如果对应f(n-3)的情况的话，从第三块砖一直不跨走到第六块砖时，这时可以再跨两块砖从第六块砖到第九块砖，而根据定义f(n-6)却没有包含这种情况，因此还需要再加上。

为了避免这样的错误，我们改正算法，对第三种情况正确分析如下：既然已经跨了两个石砖，那么下次一定是不跨砖或只跨一块砖，因此方法数为f(n-4)+f(n-5)。综上，正确的递推式为：f(n) = f(n-1)+f(n-2)+(f(n-4)+f(n-5))。在本题中，直接在main函数中利用for循环求出f(n)的值即可得到答案，时间复杂度为O(n)。

### 参考代码

/\*

Author: 林未

Result: AC Submission\_id: 319205

Created at: Sat Oct 14 2017 16:10:54 GMT+0800 (CST)

Problem\_id: 860 Time: 6 Memory: 1436

\*/

#include<cstdio>

#include<cstring>

using namespace std;

long long f[55];

int main()

{

int n;

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

{

f[0]=1;

f[1]=1;

f[2]=2;

f[3]=4;

f[4]=7;

f[5]=13;

for(int i=6; i<=50; i++)

{

f[i]=f[i-1]+f[i-2]+(f[i-4]+f[i-5]);

}

printf("%lld\n", f[n]);

}

return 0;

}

### D 水水的Horner Rule

**思路和算法分析**

本题考查的是霍纳规则。

助教非常好，将核心算法直接写在题目描述中了。霍纳规则极大缩减了多项式求值的运算时间，是非常典型的算法之一。其伪代码如下：

y = 0

for i = n downto 0

y = ai + x \* y

声明两个字符数组char x1[150], x2[150]分别记录两个H进制数。采用x11+= (long long)(x1[i]-'0')\*pow(h1,cnt1)和x22+= (long long)(x2[i]-'0')\*pow(h2,cnt2)将对应的字符数字转化为整数并按位求和计算。利用公式A(x)=a0+x(a1+x(a2+...+x(an-1+xan)···))即可求出两数对应的十进制数并得出求和的值了，时间复杂度为O(n)。

### 参考代码

/\*

Author: 林未

Result: AC Submission\_id: 317172

Created at: Fri Oct 13 2017 19:03:15 GMT+0800 (CST)

Problem\_id: 867 Time: 71 Memory: 2832

\*/

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

int main()

{

int n;

cin>>n;

while(n--)

{

long long h1, h2, x11=0, x22=0, i;

char x1[150], x2[150];

cin>>h1>>x1;

cin>>h2>>x2;

for(i=0;x1[i]!='\0';i++)

{

}

int cnt1=0;

for(i=i-1;i>=0;i--)

{

x11+=(long long)(x1[i]-'0')\*pow(h1,cnt1);

cnt1++;

}

for(i=0;x2[i]!='\0';i++)

{

}

int cnt2=0;

for(i=i-1;i>=0;i--)

{

x22+=(long long)(x2[i]-'0')\*pow(h2,cnt2);

cnt2++;

}

cout<<x11+x22<<endl;

}

return 0;

}

### E ModricWang's QuickSort

**思路和算法分析**

本题考查的是快速排序。

在上学期的数据结构课上我们已经学习了快速排序，但当时是选择第一个数作为枢轴去比较。而本题选择中间位置的数作为分隔元素，然后将小于等于它的元素放到左侧，大于它的元素放到右侧，然后对左右两侧分别进行递归操作。只需按照题中的七步描述去编写代码即可，时间复杂度为O(nlogn)。

题目要求输出第二层递归时从左往右的第二部分的元素。这就需要声明一个变量去控制递归层数，这里我声明了整型变量cnt，初值为2。经过一次递归后cnt--，当cnt==1时将数组下标为j和t间的数字按顺序输出即可。根据题意，第二层递归只需对左区间递归排序，quicksort(R,s,i)。

### 参考代码

/\*

Author: 林未

Result: AC Submission\_id: 318915

Created at: Fri Oct 13 2017 21:53:51 GMT+0800 (CST)

Problem\_id: 873 Time: 137 Memory: 5400

\*/

#include <stdio.h>

int a[1000005],n;//定义全局变量，这两个变量需要在子函数中使用

int cnt;

void quicksort(int R[],int s,int t) //对R[s]至R[t]的元素进行快速排序

{

int i=s,j=t;

int pivot;

int tmp;

pivot = R[(s+t+1)/2]; //用区间的中间位置的元素作为关键字

if (s<t) //区间内至少存在两个元素的情况

{

while (i!=j) //从区间两端交替向中间扫描,直至i=j为止

{

while (i<j && R[i]<=pivot)

i++; //从左向右扫描,找第1个大于基准记录R[i]

while (j>i && R[j]>pivot)

j--; //从右向左扫描,找第1个小于基准的R[j]

if(i<j) //将前后的两个失序元素进行交换

{

tmp=R[i];

R[i]=R[j];

R[j]=tmp;

}

}

if(cnt==2)

{

cnt--;

quicksort(R,s,i); //对左区间递归排序

}

else

{

for(int ii=j;ii<t;ii++)

printf("%d ",a[ii]);

}

}

}

int main()

{

int i,j,t;

//读入数据

cnt=2;

scanf("%d",&n);

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

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

quicksort(a,0,n-1); //快速排序调用

printf("\n");

return 0;

}

### F AlvinZH的儿时梦想——木匠篇

**思路和算法分析**

本题考查的是贪心算法。

这道题我一开始只会使用最笨的O(n2)方法，后来实在不知该如何优化便咨询了同学才掌握O(n)的算法。首先，我定义了两个数组h\_left[105], h\_right[105]去记录对应高度下左侧木条的最远距离和右侧木条的最远距离，同时声明变量hmax记录中间木条的高度。接下来，从最高的木条开始求解，如果当前高度的木条距圆心的距离比已计算过的距离x\_left / x\_right大，那么将最远距离重置为当前高度的木条距圆心的距离，此时求得的体积若比之前已计算的体积大则赋值。因为考查的木条高度在不断降低，因此当前高度i一定是正在计算的两根木条高度的较小值，也是符合题意的，这就是为什么我们要从i=100开始递减而不是从i=1开始递增的原因。同时，这也符合了高木条可以“降低姿态”与低木条的搭配装水的要求。

当然，我们也不能忘记特殊情况：**圆心处木条可看作属于两边**。因此，如果圆心处存在高度大于0的木条时，也要计算此时x\_left / x\_right与圆心处木条搭配组成的木桶能盛水多少，而对应的高度则为圆心处木条与x\_left / x\_right处木条的较小值。如果所得结果比原有记录大则赋值。最后输出结果即可。

在本题中，为防止精度丢失和简化计算，使用整型变量quantity = hx^2进行比较即可，最后得到v = quantity\*pi/4即为最终答案。

### 参考代码

/\*

Author: 林未

Result: AC Submission\_id: 319259

Created at: Sat Oct 14 2017 17:40:58 GMT+0800 (CST)

Problem\_id: 861 Time: 90 Memory: 2696

\*/

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <cmath>

using namespace std;

#define pi acos(-1)

int main()

{

int n;

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

{

int h\_left[105], h\_right[105], xx, hh;

for(int i=0; i<104; i++)

{

h\_left[i]=h\_right[i]=-1;

}

int hmax = -100;

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

{

scanf("%d %d", &xx, &hh);

if(xx>0 && xx>h\_right[hh]) h\_right[hh]=xx;

else if(xx<0 && -xx>h\_left[hh]) h\_left[hh]=-xx;

else if(xx==0 && hh>hmax) hmax = hh;

}

int x\_left=-1, x\_right=-1, quantity=0, temp;

double v;

for(int i=100; i>=1; i--)

{

if(h\_left[i]>x\_left) x\_left=h\_left[i];

if(h\_right[i]>x\_right) x\_right=h\_right[i];

// cout<<x\_left<<"\*"<<x\_right<<endl;

if(x\_left>0 && x\_right>0)

{

temp = (x\_right+x\_left)\*(x\_right+x\_left)\*i;

if(quantity<temp) quantity=temp;

}

// cout<<quantity<<endl;

if(hmax>0 && x\_left>0)

{

if(quantity<x\_left\*x\_left\*min(i, hmax))

quantity=x\_left\*x\_left\*min(i, hmax);

}

if(hmax>0 && x\_right>0)

{

if(quantity<x\_right\*x\_right\*min(i, hmax))

quantity=x\_right\*x\_right\*min(i, hmax);

}

}

//cout<<quantity<<endl;

v = quantity \* pi / 4;

printf("%.3lf\n", v);

}

return 0;

}

### G D&C--玲珑数

**思路和算法分析**

本题考查的是分治算法。

这道题的父问题是“2016级算法热身E题——模式寻对”，而它的父问题是“归并排序”。也就是说，这道题是求逆序对总数的变形题，时间复杂度为O(nlogn)。因为在Hint中已写明使用O(n2)算法会超时，因此考虑分治法求解。

首先我们回顾一下如何借助归并算法求逆序队数。方法是先将数组从中间分成两个部分，然后分别递归左半部分和右半部分，再合并排好序的左右两个部分，从而统计逆序对数。这是一次二路归并，用Merge()函数表示。比如合并两个排好序的数组{1,4,5,7}和{2,3}：（1）先比较后取出1，再比较发现2小于4，那么2肯定小于4后面的元素，从而构成了(4,2),(5,2),(7,2) 3个逆序对；（2）取出2后，再比较4和3,3<4,所以3和4以及后面的元素都构成了逆序对，共3对。利用这个方法将数组的左半边和右半边递归分解，在合并过程中统计逆序对的个数。[2]递归分解与合并的函数表示为Msort()。

在本题中，情况略微有些变化。首先将输入的数列存在数组a中。题目中对玲珑对的定义是：在数列中任意两个数a[i],a[j],如果i<j且a[i]>2\*a[j]，那么a[i]和a[j]就构成了一对儿玲珑对。因此，我们在二路归并时需要强调，如果a[i]<a[j]则正常合并，否则要判断在a[i]的状态下有没有可能出现a[i]>2\*a[j]。若有可能出现，即当定义的判断变量iii!=-1时在a[m+1~n]里寻找存不存在满足要求的a[j]。若存在这样的a[j]，则a[m+1~j]的所有元素一定均满足a[i]>2\*a[j]，因为这一段数列已有序，此时玲珑数cnt将符合要求的元素个数相加。如此递归分治即可得到最终答案。

这道题的坑点有两处。其一是p和q的大小，有可能出现p>q的情况，此时p、q要进行交换。其二是注意数据范围，如果数组a定义为int类型则只能得1/3的成绩，只有改成long long才能AC。这两点足足花了我数小时才解决，可见以后的上机还需更加细心。

### 参考代码

/\*

Author: 林未

Result: AC Submission\_id: 319198

Created at: Sat Oct 14 2017 16:04:13 GMT+0800 (CST)

Problem\_id: 869 Time: 316 Memory: 4020

\*/

#include <iostream>

#include <algorithm>

#include <cstdio>

using namespace std;

void Merge(long long SR[], long long TR[], int i, int m, int n, int &cnt)

{

if(i>=n) return ;

int j, k, ii, iii;

for(ii=m,j=n,k=n,iii=n; ii>=i&&j>=m+1; k--)

{

if(SR[ii]<=SR[j]) TR[k]=SR[j--];

else

{

if(iii!=-1)

{

for(; iii>=m+1; iii--)

{

if(SR[ii]>2\*SR[iii])

{

cnt += (iii-m);

break;

}

}

if(iii==m) iii=-1;

}

TR[k]=SR[ii--];

}

}

while(ii>=i)

{

TR[k--]=SR[ii--];

}

while(j>=m+1)

{

TR[k--]=SR[j--];

}

}

void Msort(long long SR[], long long TR1[], int s, int t, int &cnt)

{

long long TR2[t+1];

if(s==t)

{

TR1[s]=SR[s];

return ;

}

else if(s>t) return ;

else

{

int m=(s+t)/2;

Msort(SR, TR2, s, m, cnt);

Msort(SR, TR2, m+1, t, cnt);

Merge(TR2, TR1, s, m, t, cnt);

}

}

int main()

{

int n;

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

{

long long a[n];

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

{

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

}

int num;

scanf("%d", &num);

while(num--)

{

int cnt=0;

int x,y;

long long temp[n];

scanf("%d %d", &x, &y);

if(x>y) swap(x, y);

Msort(a, temp, x, y, cnt);

printf("%d\n", cnt);

}

}

return 0;

}

**三、参考资料**

[1] <http://blog.163.com/scut_yanghaoyan/blog/static/24812301320155812615279/>

[2] <http://blog.csdn.net/u010128136/article/details/52852093>