第六次练习赛题解

author: 郑耀彦，刘泽华

本题解除解方程题外均提供两份代码风格不同的代码，其中第二份代码中有解法注释。

Naïve 全排列

~~方法1：十重循环~~

此题考察递归知识，按字典序输出全排列数是非常典型的一道递归题，务必掌握！

不理解的同学可以选N=2，3，4自己按照程序走一遍过程

使用递归，第x层递归中，i从1到N循环，看是否能填入i到第x个位置(即i这个数之前没有填过，用$used$数组标记是否用过一个数)。

```
#include<stdio.h>
int used[11],n,now[11];
void dfs(int x)
{
    if(x==n+1)
    {
        int i;
        for(i=1;i<=n;i++)
            printf("%d%c",now[i],i==n?'\n':' ');
        return ;
    }
    int i;
    for(i=1;i<=n;i++)
        if(!used[i])
        {
            now[x]=i;
            used[i]=1;
            dfs(x+1);
            used[i]=0;
        }
}
int main()
{
    scanf("%d",&n);
    dfs(1);
    return 0;
}
```

```
#include <stdio.h>
#define MAX 21
int vis[MAX], path[MAX];
// vis数组用于记录数组下标对应的数字是否被使用 e.g. vis[1]=1表示1被使用; vis[1]=0表示1未被使用。
```

```c
// path数组用于记录在本次排列中数字的顺序
int N;
void fun(int n){
    int i;
    if(n > N){
    // 排列数的N个数已经全部产生
        for(i = 1; i <= N; i++)
            printf("%d ",path[i]);
        putchar('\n');
        return;
    }
    // 以下情况说明排列数还没有完全产生，需要继续递归深入产生
    for(i = 1; i <= N; i++){
    // 因为按照字典序，所以按照从小到大的顺序循环
        if(vis[i])  // 如果这个数被使用，则继续下一个数
            continue;
        vis[i] = 1; // 标记这个数被使用
        path[n] = i;
        fun(n + 1); // 继续深入找下一个数
        vis[i] = 0; // 释放这个数，表示在本次产生排列数的过程中已经用完
    }
}
int main(){
    scanf("%d", &N);
    fun(1); // 从第一个数开始产生
    return 0;
}
```

兔子序列

预备知识：减法取模的运算性质

$(a-b)\%p=(a\%p-b\%p)\%p$

$e.g.\ a=8,b=4,p=5;(a-b)\%p=(8-4)\%5=4$

数学：$(8\%5-4\%5)\%5=(3-4)\%5=(-1)\%5=4$(理由：$-1=5*(-1)+4$)

C语言：$(-1)\%5=-1$

若一对兔子在x年后开始每年繁殖一对兔子，且不停止，那么第y年的兔子数=第$y-1$年的兔子数+第$y-x+1$年的兔子数，因为第$y-x+1$年的所有兔子都能在第y年生出新兔子。如果一对兔子在z年之后不再繁殖，那么总数就要减去第$y-z+1$年的兔子总数，因为它们虽然活着但都不能繁殖了。**注意，含减法的取模运算记得将答案加回非负整数，因为序列并不递增**

```c
#include<stdio.h>
int a[111],n;
const int mod=1e9+7;
int main()
{
    a[1]=a[2]=a[3]=1;
    int i;
```

```
    for(i=4;i<=100;i++)
    {
        a[i]=(a[i-3]+a[i-1])%mod;
        if(i>=8)a[i]=(a[i]-a[i-7])%mod;
    }
    int n;
    scanf("%d",&n);
    printf("%d",a[n]<0?a[n]+mod:a[n]);
    return 0;
}
```

```
#include <stdio.h>
#define mod 1000000007
int main() {
    int i, N;
    scanf("%d", &N);
    long long n[10] = {0, 1, 1, 1, 2};
    // 大家联系斐波那契数列产生过程，斐波那契数列的产生过程中需要记录三个数（前两个数和当前数），本题需要记录的数比斐波那契数列多了几个
    for(i = 5; i <= 7; i++)
        n[i] = n[i - 1] + n[i - 3];
    if(N <= 7)
        printf("%lld", n[N]);
    // N <= 7 的过程中没有不再繁殖的兔子，此为一种情况
    // 从第八年开始有不再繁殖的兔子，此为另一种情况
    else {
        N -= 7;
        while(N--) {
        // 与产生斐波那契数列过程类似
        // 本年的兔子数等于前一年的兔子数加上即将要新产生的兔子数
            n[8] = (n[7] + n[5] - n[1]) % mod;
            if(n[8] < 0)
                n[8] += mod;
            for(i = 1; i <= 7; i++)
                n[i] = n[i + 1];
        }
        printf("%lld", n[7]);
    }
    return 0;
}
```

高精度减法

按字符串输入进来转换为整数数组，为了对齐低位，将数组翻转，然后用小学学的竖式减法。可以先判断一下大小，顺便特判$A = B$。

```
#include<stdio.h>
#include<string.h>
void change2num(char *s,int *a,int len)
```

```
{
    int i;
    for(i=1;i<=len;i++)a[i]=s[i]-'0';
}
void reverse(int *a,int len)
{
    int i;
    for(i=1;i<=len/2;i++)
    {
        int temp=a[i];
        a[i]=a[len-i+1];
        a[len-i+1]=temp;
    }
}
int cmp(char *s,char *t,int lens,int lent)
{
    if(lens>lent)return 1;
    if(lens<lent)return 0;
    int i;
    for(i=1;i<=lens;i++)
    {
        if(s[i]>t[i])return 1;
        if(s[i]<t[i])return 0;
    }
    return -1;
}
char s[111],t[111];
int a[111],b[111],c[111];
int main()
{
    scanf("%s%s",s+1,t+1);
    int lens=strlen(s+1),lent=strlen(t+1);
    int bigger=cmp(s,t,lens,lent);
    if(bigger==-1)
    {
        printf("0");
        return 0;
    }
    int i;
    change2num(s,a,lens);
    change2num(t,b,lent);
    reverse(a,lens);
    reverse(b,lent);
    int len=lens>lent?lens:lent;
    if(!bigger)
        for(i=1;i<=len;i++)
        {
            int temp=a[i];
            a[i]=b[i];
            b[i]=temp;
        }
    //calc
    int ex=0;
```

```
    for(i=1;i<=len;i++)
    {
        c[i]=a[i]-b[i]+ex;
        if(c[i]<0)ex=-1,c[i]+=10;
        else ex=0;
    }
    //output
    if(!bigger)printf("-");
    int lenc=len;
    while(c[lenc]==0)lenc--;
    for(i=lenc;i>=1;i--)printf("%d",c[i]);
    return 0;
}
```

```
#include <stdio.h>
#include <string.h>
#define MAX 103
int cmp(char A[], char B[]);    // cmp函数用于判断A和B两个“数”的大小
void sub(char A[], char B[]);   // 减法，“大数”减“小数”
int main(){
    char A[MAX], B[MAX];
    scanf("%s", A);
    scanf("%s", B);
    int key = cmp(A, B);
    // cmp函数返回值0表示两数相等；返回值1表示A大于B；返回值-1表示A小于B；
    if(key == 1){
        sub(A, B);
    } else if(key == -1){
        putchar('-');
        sub(B, A);
    } else {
        printf("0");
    }
    return 0;
}
int cmp(char A[], char B[]){
    // 首先通过长度判断
    int len1, len2;
    len1 = strlen(A);
    len2 = strlen(B);
    if(len1 > len2) {
        return 1;
    } else if(len1 < len2) {
        return -1;
    } else{
    // 如果两个数长度相等，判断从高位开始第一个不同的数
        int i;
        for(i = 0; i < len1; i++)
            if(A[i] > B[i])
                return 1;
            else if(A[i] < B[i])
                return -1;
            else
```

```
                continue;
        return 0;   // A == B
    }
}
void sub(char A[], char B[]){
    int len1, len2;
    len1 = strlen(A);
    len2 = strlen(B);
    int i, j, tmp;
    int a[MAX], b[MAX], ans[MAX] = {0};
    // 减法和加法都是从低位开始运算，而我们的输入是数的高位对应数组下标小的元素，低位对应数组下标大的元素；因此按照习惯将两个字符串反转（顺便将字符处理为数字）
    for(i = 0, j = len1 - 1; i <= j; i++, j--){
        a[i] = A[j] - '0';
        a[j] = A[i] - '0';
    }
    for(i = 0, j = len2 - 1; i <= j; i++, j--){
        b[i] = B[j] - '0';
        b[j] = B[i] - '0';
    }
    for(i = 0; i < len2; i++){
    // 以字符串短的为标准，模拟竖式运算做减法，不够从高位借
        tmp = a[i] - b[i];
        if(tmp < 0){
            ans[i] = tmp + 10;
            a[i + 1] -= 1;
        } else {
            ans[i] = tmp;
        }
    }
    // 不再进行减法，将大数的剩余位补足到答案
    // 注意：不能直接从将a数组元素赋值给ans，因为可能存在上面的运算后从高位借位的情况
    // e.g. 100000 - 999 = 99001,  而不是100001
    for(i = len2; i < len1; i++){
        if(a[i] < 0){
            ans[i] = a[i] + 10;
            a[i + 1] -= 1;
        } else{
            ans[i] = a[i];
        }
    }
    while(1){
    // 从高位开始找到第一个不为0的元素
    // 前提：因为A! =B，所以必定存在一个不为0的元素；如果这点不能保证，则会产生数组越界的情况（数组下标一直减到负数）
        if(ans[i] != 0)
            break;
        i--;
    }
    for(i; i >= 0; i--)
        printf("%d",ans[i]);
}
```

login学线代

高斯消元，把第i行的前$i-1$个数消成0，最后倒着求出答案。由于保证有唯一解，所以特殊情况不多。

```
#include<stdio.h>
#include<math.h>
double a[111][111];
double x[111];
const double eps=1e-13;
int dcmp(double x)
{
    if(fabs(x)<eps)return 0;
    if(x<0)return -1;
    return 1;
}
int main()
{
    int n;
    scanf("%d",&n);
    int i,j;
    for(i=1;i<=n;i++)
        for(j=1;j<=n+1;j++)scanf("%lf",&a[i][j]);
    for(i=1;i<=n;i++)
    {
        int temp,j;
        if(dcmp(a[i][i]==0))//第i行第i个数为0的话，在后面的行找一个第i个数不为0的行，将其换到第i
行
        {
            for(j=i+1;j<=n;j++)
                if(dcmp(a[j][i]))
                {
                    temp=j;
                    break;
                }
            for(j=1;j<=n+1;j++)
            {
                double t=a[i][j];
                a[i][j]=a[temp][j];
                a[temp][j]=t;
            }
        }
        for(j=i+1;j<=n;j++)if(dcmp(a[j][i]))//用第i行将所有后面的行的第i个数消为0
        {
            double bi=a[i][i]/a[j][i];
            int k;
            for(k=i;k<=n+1;k++)
                a[j][k]=a[j][k]*bi-a[i][k];
        }
    }
    for(i=n;i>=1;i--)//从后往前算x
    {
```

```
        x[i]=a[i][n+1]/a[i][i];
        for(j=i-1;j>=1;j--)
            a[j][n+1]-=a[j][i]*x[i];
    }
    for(i=1;i<=n;i++)printf("%.10f\n",x[i]);
    return 0;
}
```

酸奶想成为魔法少女5

按部就班地翻就完事了

```
#include<stdio.h>
char s[1111];
int main()
{
    scanf("%s",s);
    int n,i;
    scanf("%d",&n);
    while(n--)
    {
        int l,r;
        scanf("%d%d",&l,&r);
        for(i=0;i<(r-l+1)/2;i++)
        {
            char ch=s[l+i];
            s[l+i]=s[r-i];
            s[r-i]=ch;
        }
    }
    printf("%s",s);
    return 0;
}
```

```
#include <stdio.h>
#include <string.h>
#define MAX 1001    // 定义字符串最大长度
char s[MAX];    // 定义的函数直接对全局变量改变
void swap(int l, int r){
// 定义交换函数每次交换下标从l到r之间的字符串部分
    int i, j;
    char temp;
    for(i = l, j = r; i < j; i++, j--){
        temp = s[i];
        s[i] = s[j];
        s[j] = temp;
    }
}
int main()
{
    int n, l, r;
```

```
    scanf("%s", s);
    scanf("%d", &n);
    while(n--){
    // n组数据，共交换n次
        scanf("%d%d", &l, &r);
        swap(l, r);
    }
    printf("%s", s);
    return 0;
}
```