士谔2019第7次练习赛题解

期末考试很快就要到了，这里提供几条复习建议

1. 最后几次练习赛题目较为复杂，可以参考题解，思考如何使用简洁的方法编写程序。
2. 每天做题，保持手感。
3. 总结以前提交未通过时的错误，考试时评测机评测时间可能较长，争取一次AC。
4. 学会使用断点调试，这对Debug很有帮助
5. 可以带纸质资料，做必要的笔记标注，如一些库函数的定义等。

A.多简单

解题思路

本题是一道很基础的指针应用，之前上课和上机中大家都接触过*swap*函数，和本题中*func*函数相同，对于函数内数字的计算和交换等操作，如果不依赖返回值，想让它在函数中计算结果得到保留，那就必须使用指针。如果对这方面仍然不是很了解，建议去查看*ppt*或者教材再温习一下，指针是*c*语言的精髓（也是下学期数据结构的重点之一）。

对于计算，按题面描述计算即可。最后的表达式是数学逻辑推理出来的，在标准代码中给出。

AC代码

```
#include <stdio.h>
void fun(int *a,int *b){
    *a = *a + *b;
    *b = *a - *b;
}
int main()
{
    int a,b,c,d,na,nb,nc,nd;
    scanf("%d%d%d%d",&a,&b,&c,&d);
    na=a,nb=b,nc=c,nd=d;
    fun(&na,&nb);
    fun(&nc,&nd);
    fun(&nb,&nc);
    printf("%d %d %d %d\n",na,nb,nc,nd);
    printf("na=a+b\n");
    printf("nb=a+c+d\n");
    printf("nc=a\n");
    printf("nd=c");
    return 0;
}
```

B. lyd的IP地址

解题思路

原样输出一定要复制，不然想半天WA哪里结果发现是`Inva1id`打成了`Invalid`。

下方提供了两种解题思路，分别是逐字符处理和使用库函数快速处理。

逐字符处理：对于每一行的输入，如果它拥有`.`就代表它可能是`IPv4`地址，如果它拥有`:`，就代表它可能是`IPv6`地址。对于`IPv4`地址，它应该拥有三个`.`，且被`.`分割的四个空间的字符数为1~3，数值为0~255。对于`IPv6`地址，它应该拥有七个`:`，且被`:`分割的八个空间的字符数为1~4，数值为0~0xFFFF。也就是说逐字符处理对于每行有三次检测，分别是有几个分割的`.`或`:`，每个空间有几个字符，数值是否在允许的范围内。

库函数处理：在判断是`IPv4`还是`IPv6`地址后，使用 $sscanf$ 函数从字符串中读取数字，如果读取成功的变量数为4或者8，并且每个变量都在允许的范围内，则为有效的`IPv4`或`IPv6`地址，否则输出`Inva1id`。

AC代码1 库函数处理

```
#include<stdio.h>
int main()
{
    int a,b,c,d;
    char s[2048],t[2048];
    while(gets(s))
        //使用sscanf格式控制进行匹配
        if(sscanf(s, "%4x:%4x:%4x:%4x:%4x:%4x:%4x:%4x%s",
                    &a,&a,&a,&a,&a,&a,&a,&a,s)==8)
            puts("IPv6");
        else
            puts(sscanf(s, "%3d.%3d.%3d.%3d%s", &a,&b,&c,&d,t)==4 && (a|b|c|d)
<256 ?                       "IPv4":"Inva1id");
}
```

AC代码2 逐字符处理

链接 https://cppaste.com/p/DkibCaR0

C. 来个质数（地狱版）

解题思路

本题实现三个函数即可轻松解决：

1. 判断n是否为质数，这里使用从2遍历到$\sqrt{n}$的方法，逐个检查该数字是否为n的约数。
2. 求n的最小非2质因数。之前我们写过对n进行分解质因数的操作，现在只需要从2开始递增查找，找到第一个非2质因数返回就行。
3. 求n的最大质因数。其实就是进行分解质因数的整个过程，最后那个质因数就是最大的质因数。

有了这三个函数，剩下的工作就是严格按照题意进行操作了。

AC代码

```
#include <stdio.h>
#include <string.h>

int is_prime(int n) {
    int i;
    for (i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}

int get_min_factor(int n) {
    int i;
    for (i = 2; i <= n; i++) {
        if (n % i == 0) {
            if (i != 2) return i;
            n /= i;
            i--;
        }
    }
    return 0;
}

int get_max_factor(int n) {
    int i, max = 0;
    for (i = 2; i <= n; i++) {
        if (n % i == 0) {
            max = i;
            n /= i;
            i--;
        }
    }
    return max;
}

int main() {
    char s[1000];
    int i, len, n = 0;
    scanf("%s", s);
```

```
    len = strlen(s);
    for (i = 0; i < len; i++) {
        if (s[i] == ',') continue;
        n = n * 2 + s[i] - '0';
    }
    if (n <= 5) printf("%d", n);
    else {
        if (is_prime(n)) {
            int p = n;
            do {
                int min_factor = get_min_factor(p + 1);
                p = min_factor == 0 ? 5 : min_factor;
            } while (p >= 10);
            printf("%d", n * p / 10);
        }
        else {
            int max_factor = get_max_factor(n);
            int price = n;
            while (price >= 0.9 * n) {
                price -= max_factor;
            }
            printf("%d", price);
        }
    }
    return 0;
}
```

D. yuki舒的子序列

题解思路

问无限次循环出现字符串s的字符串最早在什么位置能使得t是它的子序列。

显然，当且仅当t中出现过s中未出现过的字母时输出`-1`，可以用一个数组来记录每个字母是否在s中出现过，再对t中的字母逐个判断。

对于可以成为子序列的情形，我们用一个数组`nxtdis[i][j]`来表示对于s中的位置i下一个字母j在它多少个字符之后。然后就可以利用该数组依次得出到t的每一个位置s的循环串所匹配到的位置。

`nxtdis[i][j]`可以在对s从后往前扫的过程中求得。（详见代码，以下代码中的实现是将一个s拼接在s后以便统计）。

AC代码

```
#include<stdio.h>
#include<string.h>
char s[200005],t[100005];
int hav[26],pos[26],nxtdis[100005][26];
int main()
{
    scanf("%s%s",s,t);
    int lens=strlen(s),lent=strlen(t);
    for(int i=0;i<lens;i++)hav[s[i]-'a']++,s[i+lens]=s[i];
    for(int i=0;i<lent;i++)if(!hav[t[i]-'a']){puts("-1");return 0;}
    for(int i=2*lens-1;i>=0;i--)
    {
        if(i<lens)
        for(int j=0;j<26;j++)nxtdis[i][j]=pos[j]-i;
        pos[s[i]-'a']=i;
    }
    long long ans=(t[0]==s[0])?0:nxtdis[0][t[0]-'a'];
    for(int i=1;i<lent;i++)ans+=nxtdis[ans%lens][t[i]-'a'];
    printf("%lld\n",ans+1);
    return 0;
}
```

E. 毛毛虫和孔雀

解题思路

如果对于每一个询问都遍历一遍数组进行查询会超时，考虑到已排序，使用二分优化查询上下界。

AC代码

```
#include <stdio.h>
#define N 100005
int len[N], n, q;
//第一个长度小于k的位置+1，超出范围返回-1
int findLow(int k){
    //考虑边界情况，r最小为0，根据终止条件则l为-1
    int l = -1, r = n -1 ;
    if(len[0] > k || len[n-1] < k)return -1;
    //保障len[r]始终大于等于k，len[l]始终小于k
    //l为第一个长度小于k的位置时终止。此时r=l+1
    while(l < r - 1){
        int mid = (l + r) / 2;
        //相等时仍需调整上界，使最终r为第一个k的位置
        if(len[mid] >= k)r = mid;
        else l = mid;
    }
    return r;
}
//第一个长度大于k的位置-1，超出范围返回-1
int findHigh(int k){
    int l = 0, r = n;
    if(len[0] > k || len[n-1] < k)return -1;
    while(l < r - 1){
        int mid = (l + r) / 2;
        if(len[mid] <= k)l = mid;
        else r = mid;
    }
    return l;
}
int main(){
    scanf("%d %d", &n, &q);
    int i, j;
    for (i = 0; i < n; ++i) {
        scanf("%d", len + i);
    }
    for (j = 0; j < q; ++j) {
        int x = 0;
        scanf("%d", &x);
        int l = findLow(x), r = findHigh(x);
        if(l <= r && l != -1 && r != -1) printf("%d %d\n", l + 1, r + 1);
        else printf("-1\n");
    }
}
```

此题也可以使用调整查询顺序、使用哈希函数等方法进行优化。

F.面基

解题思路

本题我们主要考察直角坐标系中已知三点坐标如何**准确**求解三角形面积。首先我们知道，一个简单n多边形可以选择一个点为基准点，然后按照顺时针或逆时针的顺序将它连续切分成$n-2$个三角形。就本题而言，不妨设输入的第1个点为基准点。那么对应的$n-2$个三角形就是$S(1,2,3),S(1,3,4),\cdots,S(1,n-1,n)$.另一方面，相信正在学习高等代数的同学们一定知道已知三点坐标时求解三角形**有向**面积的公式为

$$S=\frac{1}{2}\begin{vmatrix}x_1 & y_2 & 1\\ x_2 & y_2 & 1\\ x_3 & y_3 & 1\end{vmatrix}$$

将该式展开即可得到用代码表示的面积公式。

同时需要注意的是，由于输入的多边形可能是**凹多边形**，所以正确的做法是把所有三角形的有向面积相加后再取绝对值，否则可能会WA。

当然本题也有其他计算面积的方式，欢迎同学们自行研究。

参考代码

```
#include <stdio.h>
#include <math.h>
double basex,basey;
double size(double curx,double cury,double lastx,double lasty){
    return (curx*lasty+lastx*basey+basex*cury-basex*lasty-lastx*cury-
curx*basey)/2.0;
}
int main(){
    int n;
    double curx,cury,lastx,lasty;
    scanf("%d",&n);
    scanf("%lf %lf",&basex,&basey);
    scanf("%lf %lf",&lastx,&lasty);
    n-=2;
    double sum = 0;
    while(n--){
        scanf("%lf %lf",&curx,&cury);
        sum += size(curx,cury,lastx,lasty);
        lastx = curx;
        lasty = cury;
    }
    printf("%.7f",((sum>0)?sum:-sum));
}
```

G.汉明帝斯坦斯

解题思路

对于每组数据，如果满足要求，当且仅当满足下列两个条件：
1）每个字符串中字母的种类、数目完全相同。
2）存在一个字符串使任意一个字符串和该串的汉明距离小于等于2。汉明距离是一个概念，它表示两个（相同长度）字对应位不同的数量，我们以d（x,y）表示两个字x,y之间的汉明距离。

```
#include<stdio.h>
char a[5010][5010];
int num[5010][26],dif[5010];
int main(){
    int i,j,k,n,l,T,x,y;
    scanf("%d",&T);
    while(T--){
        int same=0;
        memset(num,0,sizeof(num));
        memset(dif,0,sizeof(dif));
        scanf("%d%d",&n,&l);
        for(i=0;i<n;i++){
            scanf("%s",a[i]);
            for(j=0;j<l;j++)num[i][a[i][j]-'a']++;
        }
        int flag=0;
        for(i=1;i<n&&!flag;i++){
            for(j=0;j<26;j++){
                if(num[i][j]>1)same=1;
                if(num[i][j]!=num[0][j])flag=1;
            }
        }
        if(flag){
            printf("-1\n");
            continue;
        }
        for(i=1;i<n;i++)
            for(j=0;j<l;j++)
                if(a[0][j]!=a[i][j])dif[i]++;
        for(i=0;i<l&&!flag;i++){
            for(j=i+1;j<l;j++){
                int can=1;
                for(k=1;k<n;k++){
                    int diff=dif[k]-(a[0][i]!=a[k][i])-(a[0][j]!=a[k][j])+(a[0]
[i]!=a[k][j])+(a[0][j]!=a[k][i]);
                    if((diff==0&&same)||diff==2);else can=0;
                }
                if(can){
                    flag=1;
                    char t=a[0][i];a[0][i]=a[0][j];a[0][j]=t;
                    printf("%s\n",a[0]);
                    break;
                }
            }
        }
```

```
        if(!flag)printf("-1\n");
    }
    return 0;
}
```