周赛一题解

A、防抖

出题人：蔡郭平

我们可以发现，当一个请求发出后t秒内未有下一个请求时，该请求就可以被满足，执行时间为a_i+t。所以我们只需要依次判断每个请求与上一个请求的间隔时间即可。

最后一个请求则是必然会被执行的。

注：因为OJ的问题，本次A题输出行末回车会导致【格式错误】。在此向所有参赛者谢罪，但也提醒各位参赛者一定要注意查看比赛群的公告，得知实时的信息。

```
#include <stdio.h>
#define ll long long
int a[10005];
int ans[10005];
int main()
{
    int n,t;
    scanf("%d%d",&n,&t);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    int cnt=0;
    for(int i=2;i<=n;i++)
        if(a[i]-a[i-1]>=t) ans[++cnt]=a[i-1]+t;
    ans[++cnt]=a[n]+t;
    printf("%d\n",cnt);
    for(int i=1;i<=cnt;i++)
        printf("%d ",ans[i]);
    return 0;
}
```

B、蕴含链

出题人：高言峰

注意到，一旦有一个1，就可以把它放在最后，那么整体的结果就是1。

那么只需要考虑全为0的情形。注意，全为0时，每多一个0，蕴含链的结果就会改变一次，而只有一个0时结果是0。因此由数学归纳法可证，当0的个数是奇数时，结果为0；当0的个数是偶数时，结果为0。

```
#include <stdio.h>
int main()
```

```
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        int n;
        scanf("%d", &n);
        int ok = n % 2 == 0;
        while (n--)
        {
            int x;
            scanf("%d", &x);
            if (x == 1)
                ok = 1;
        }
        puts(ok ? "YES" : "NO");
    }
    return 0;
}
```

C、偶回文串

出题人：李嘉龙

因为给的字符串的长度比较长，所以不能依次枚举区间$[i,j]$的子串看它是不是偶回文串，这样是会超时的。

比较好的方法是设$d[i]$表示以第i个元素作为偶回文串中心时，有多少个偶回文串。

如果$s[i]=s[i+1]$，那么$d[i]=1$。

如果与此同时$s[i-1]=s[i+2]$，那么$d[i]=2$.....

以此类推，对于每个i都不断地拓展到字符串的边界。

而且能保证，这样的枚举不会枚举到重复的偶回文串。因为每个偶回文串要么中心不同，要么长度不同。

```
#include <stdio.h>
#include <string.h>
char s[1010];
int d[1010];
int main()
{
    int n;
    scanf("%d ",&n);
    while(n--)
    {
        memset(d,0,sizeof(d));
        gets(s);
        int len=strlen(s);
```

```
        int i;
        int ans=0;
        for(i=0;i<len;i++)
        {
            d[i]=0;//初始时数量为0
            while(i-d[i]>=0&&i+d[i]+1<len&&s[i-d[i]]==s[i+d[i]+1])
d[i]++;
            //在字符串的边界内往两边拓展，看看能整出几个偶的回文串
            ans+=d[i];
        }
        if(ans!=0) printf("%d\n",ans);
        else printf("HelenCai!\n");
    }
    return 0;
}
```

如果字符串的长度继续变长，那就涉及到Manacher算法了，属于超纲内容，有兴趣者可以自行了解。

D、Disco Elysium II

出题人：李泽政

本题的题目要求其实就是让我们找到a和b。

a的可能范围很小，在$[1,32767]$区间内，而且是整数，所以我们可以枚举a。

而且对于每个a，因为b小于32768，所以其对应的b是只有一个的。

为什么呢？因为在a不变的情况下想要生成同样的下一种子只可能：

$(seed*a+b)\%p=(seed*a+b+p)\%p$。

而$b+p$显然是大于32768，不满足其给的范围。

我们用前两个种子由a得到b，需要注意b不能是负数。

对于每组枚举出来的a和b，我们都去验证一遍生成的n个数满不满足，因为题目提示了不存在多解，所以如果找到一组满足的a,b，就是答案。

如果枚举完了仍旧没能找到满足的一组，那就是无解。

需要注意的是：

不能既枚举a的同时又去枚举b，32768*32768的数据量会超时。

```
#include <stdio.h>
#define ll long long
int x[105];
int b[40000];
int main()
```

```
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n;
        scanf("%d",&n);
        for(int i=1;i<=n;i++) scanf("%d",&x[i]);
        for(int i=1;i<=32767;i++)//找到每个a对应b
        {
            b[i]=x[2]-x[1]*i%32768;
            while(b[i]<0) b[i]+=32768;
        }
        int f=0;
        int ans=0;
        for(int i=1;i<=32767;i++)
        {
            int cnt=2;
            while(cnt!=n)
            {
                if(((x[cnt]*i+b[i])%32768)!=x[cnt+1]) break;
                cnt++;
            }
            if(cnt==n)//找到一组满足的
            {
                ans=(x[n]*i+b[i])%32768;
                f=1;
                break;
            }
        }
        if(f) printf("%d\n",ans);
        else printf("No Solution!\n");
    }
    return 0;
}
```

如果本题的p进一步扩大，那本题就将涉及数论方面的知识.....有兴趣者可以自行了解。想得到这种解法标程的可以私下询问高言峰学长。

我们注意到有部分有基础的同学试图使用数论方面的算法来解决该题，我们必须要提醒这些同学，p不是质数，不互质的情况下是求不了逆元的。

E、方格染色游戏

出题人：李嘉龙

这题属于超纲范围，因为考到了较为简单的动态规划，也正因如此放在了最后一题。

我们可以设$dp[i][j][e][f]$表示染了i块红色，j块蓝色的情况下，任意长度的连续方格里，红色-蓝色最多为e，蓝色-红色最多为f的可能方案数。

则：

如果眼下的这块方格染成红色：

$dp[i+1][j][e+1][max(f-1,0)]+=dp[i][j][e][f]$。

如果眼下的这块方格染成蓝色：

$dp[i][j+1][max(e-1,0)][f+1]+=dp[i][j][e][f]$。

然后$dp[0][0][0][0]=1$，一步步往上递推即可。

最后答案是$dp[a][b][x][y],(x\leq 9,y\leq 9)$的总和。

```
#include <stdio.h>
#define ll long long
const int maxn=150+10;
ll mod=114514;
int dp[maxn][maxn][25][25];
int max(int a,int b)
{
    return a>b?a:b;
}
int main()
{
    int n,a,b,k;
    scanf("%d%d%d",&n,&a,&b);
    k=9;
    dp[0][0][0][0]=1;
    for(int i=0;i<=a;i++)
        for(int j=0;j<=b;j++)
            for(int e=0;e<=k;e++)
                for(int f=0;f<=k;f++)
                {
                    int now=dp[i][j][e][f];
                    (dp[i+1][j][e+1][max(f-1,0)]+=now)%=mod;
                    (dp[i][j+1][max(e-1,0)][f+1]+=now)%=mod;
                }
    ll ans=0;
    for(int e=0;e<=k;e++)
        for(int f=0;f<=k;f++) (ans+=dp[a][b][e][f])%=mod;
    printf("%lld\n",ans);
    return 0;
}
```

总结：

A题考察了读题能力，但因OJ问题有了一点锅，也警示了参赛者要接受群内信息。

B题考察了思维的灵活，以及一些细节是否有想到。

C题考察了字符串的简单处理，以及思维角度的切换（不是考虑回文串的开始，而是回文串的中心）。

D题考察了简单的数学知识，但对思维灵活的要求较高，如果能想到每个a都只对应一个b，思维能力应该很不错。

E题考察到了对动态规划的理解，属于本次周赛唯一超纲题，对于没有基础的同学有一点艰难，主要用处是防AK。

总的来说，本次周赛的特点是：思维量大，代码量小，且ABCD四题是无需用到任何算法的题目。

下次周赛预计将会与本次周赛的特点略有不同，敬请期待。