# 肖志昊的20221114周考试总结

> Never gonna give you up, never gonna let you down
> Never gonna run around and desert you
> Never gonna make you cry, never gonna say goodbye
> Never gonna tell a lie and hurt you ——Rick Astley



| | 11.14 | 预期得分 | 实际得分 |
|---|---|---|---|
| $T1$ | $substr.cpp$ | $0pts$ | $0pts$ |
| $T2$ | $flower.cpp$ | $40pts$ | $40pts$ |
| $T3$ | $refract.cpp$ | $0pts$ | $0pts$ |
| $T4$ | $paint.cpp$ | $?\ pts$ | $8pts$ |

| | 11.15 | 预期得分 | 实际得分 |
|---|---|---|---|
| $T1$ | $reverse.cpp$ | $60pts$ | $61pts$ |
| $T2$ | $silhouette.cpp$ | $20pts$ | $10pts$ |
| $T3$ | $seat.cpp$ | $8pts$ | $8pts$ |
| $T4$ | $ancient.cpp$ | $20pts$ | $5pts$ |

这两次考试都是大寄啊.....分析下原因

## D1T1 - substr.cpp

一眼配对（首尾相接，一个首对应一个尾），鉴定为纯纯的费用流。

但是咱的建模能力还是太弱了！一开始并没有看到这个类似于二分图（每个字符串本身 / 它作为前缀）的模型，所以建立了一个错误的模型，遇到了负环，于是为了解决负环，魔改费用流魔改了许久，结果没有成功，时间花费了很多，间接导致了 $T3$ 没有时间看。最后在代码收上去后才写完了这个费用流的代码（于是考试得分 $0pts$），得分 $40pts$ ，事实上不如全排列的暴力 $50pts$，但是如果对于小数据使用全排列，大数据使用费用流的话，可以得分 $60pts$。好嘛，其实看起来也不是很赚（写个费用流拿 $10pts$）。

费用流可以求解本题的“使总长度最短的长度”是多少，也可以通过边的流量来构造出答案字符串，但是无法解决字典序最小。于是我们搬来了今天的正解——AC自动机。

这里先具体介绍下模板的AC自动机（咱今天才整完理解完并且敲了 $1+1+4-5-1+4$ 遍，耶）。它本身可以理解为一棵加上了失配指针的Trie。其核心语句如下：（首先我们使用bfs，使得保证搜索到这个节点的时候其失配指针已经处理出来）

```
while(!q.empty()){
    const int u = q.front();
    q.pop();
    for(int i = 0;i < 26;i++){
        if(tr[u][i]){
            fail[tr[u][i]] = tr[fail[u]][i];
            //stat[tr[u][i]] |= stat[fail[tr[u][i]]]; //魔改代码
            q.push(tr[u][i]);
        }else{
            tr[u][i] = tr[fail[u]][i];
        }
    }
}
```

我们可以进行一个分类讨论来考虑：

- 如果该节点有"i儿子"，那么“i儿子”的失配指针就是自己的失配指针的“i儿子”（可以看做：把自己的儿子的失配转换为自己的失配）
- 如果该节点没有“i儿子”，那么访问到这里就已经失配的，所以直接指向失配指针的“i儿子”

于是我们就写好了一个可以匹配多字符串的AC自动机。而对于本题，我们需要魔改一下：对于这个trie，我们用其维护一个状压后的状态——表示当前那些串已经出现完了。有两种情况该字符串已经出现完了：

1. 我们搜索到的节点为某个字符串的结尾对应的节点
2. 我们搜索到的节点是通过失配搜索到的，也就是说该字符串没有搜索完就跳到了另外一个字符串，通过AC自动机的失配指针可以得知：这相当于把两个字符串接起来，删除共同部分

于是我们最后进行一个搜索就可以找到答案了。

# 代码

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 2006;
const int st = 0;
int n,tnt = st;
string instr;
int tr[MAXN][26],fail[MAXN],stat[MAXN];
void insert(string str,int no){
    int u = st,len = str.length() - 1;
```

```
    for(int i = 1;i <= len;i++){
        const int c = str[i] - 'A';
        if(!tr[u][c])tr[u][c] = ++tnt;
        u = tr[u][c];
    }
    stat[u] |= (1 << (no-1));
    return ;
}
queue<int> qb;
void build(){
    for(int i = 0;i < 26;i++)if(tr[st][i] != 0)qb.push(tr[st][i]);
    while(!qb.empty()){
        const int u = qb.front();
        qb.pop();
        for(int i = 0;i < 26;i++){
            if(tr[u][i]){
                fail[tr[u][i]] = tr[fail[u]][i];
                stat[tr[u][i]] |= stat[fail[tr[u][i]]];
                qb.push(tr[u][i]);
            }else{
                tr[u][i] = tr[fail[u]][i];
            }
        }
    }
    return ;
}
int output_tail,outputs[MAXN*(1<<12) + 5];
int head = 1,tail = 1;
struct sq{
    int pos,lst,stats;
    char ch;
}q[MAXN*(1<<12) + 5],csq;
void output_ans(int u){
    output_tail = 0;
    for(;u;u = q[u].lst){
        outputs[++output_tail] = q[u].ch;
    }
    for(int i = output_tail - 1;i >= 1;i--){
        putchar('A' + outputs[i]);
    }
    return ;
}
bool vis[MAXN][1<<13 + 2];
void bfs(){
    q[1].stats = 0;
    q[1].pos = st;
    q[1].lst = 0;
    q[1].ch = 0;
    while(head <= tail){
        const int u = q[head].pos,chr = q[head].ch,nows = q[head].stats;
        if(nows == (1 << n) - 1){
            output_ans(head);
            return ;
        }
        head++;
```

```
        for(int i = 0;i < 26;i++){
            csq.pos = tr[u][i],csq.ch = i,csq.lst = head - 1;
            csq.stats = nows | stat[csq.pos];
            if(!vis[csq.pos][csq.stats]){
                vis[csq.pos][csq.stats] = 1;
                q[++tail] = csq;
            }
        }
    }
    return ;
}
int main(){
    freopen("substr.in","r",stdin);
    freopen("substr.out","w",stdout);
    cin>>n;
    for(int i = 1;i <= n;i++){
        cin>>instr;
        insert("." + instr,i);
    }
    build();
    bfs();
    return 0;
}
```

# D1T2 - flower.cpp

一眼区间，鉴定为莫队（才学完莫队学魔怔了）

然而事实上模数在变，无法离线。后来才发现咱没有关注到 $k$ 的范围只有 $10^5$ 啊！于是我们可以分别对于每一个可能的 $k$ 值暴力分块加上预处理来解决整块的问题，剩下的就分块处理了。（目前未改出正解，不过这道题还是很可改的）

# D1T3 - refract.cpp

一眼...考试的时候没时间，看都没有看（

正解是一个诡异的DP（？伪DP？），绝赞理解中...

# D1T4 - paint.cpp

一眼不会做，鉴定为`puts("2");`。

然而事实上，判断黑色块的连通块个数能够拿 $41pts$ ！可以说是亏爆了。

我们可以得出：每个黑色块单独染色一定是可行的（不一定最优），那么输出黑色块连通块数量是可以骗一定的分的。

所以咱学到了——有“智慧一点”的骗分的时候应该写“智慧一点”的骗分。

# D2T1 - reverse.cpp

一眼字符串，鉴定为不会做——因为按照咱的理解来说，这个样例是错误的啊！！！

于是跳过了这道题，最后写完了其他题才来补 $T1$。然后发现我们理解错了**翻转(reverse)**这个词，它的意思并不是 $0/1$ 翻转，而是类似于`reverse(s.begin(),s.end())`，表示的是坐标沿中轴线对称。于是我们可以发现，始终只有一个 $1$，那么咱就想到了`bfs`。于是拿到了 $61pts$，但是这并不全都是因为暴力而T掉了。我们在交换坐标的时候没有考虑到字符串的范围是否在 $[1,n]$ 内，于是会导致步数变少（走了一些本来不可能的路径），修改后 $81pts$ ，剩下的点TLE。

考虑优化这个BFS，我们采取的是使用两个`set`表示剩下的没有选取的点的下标，因为我们之前的 $O(nk)$ 复杂度主要在于枚举到了很多无用的点，两个`set`配合上`lower_bound`可以把这个 $k$ 化为 $log\ n$。于是就解决了本题（分奇偶是因为如果一个可到的下标为奇，剩下的都是奇，反之同理）。

# 代码

从这天开始尝试了新的缺省源~

```
#include<bits/stdc++.h>
#define FILE_ST using namespace std;namespace Mi_Shatas_Lilion{
#define EXCUTE_ST int main(){
#define FILE_ED_And_excute return 0;}}int main(){return
Mi_Shatas_Lilion::main();}
#pragma GCC optimize(2)
FILE_ST
const int MAXN = 100006;
int n,k,m,s,ans[MAXN];
bool est[MAXN];
queue<int> q;
set<int> set1,set2;
inline int read(){
    //read快读
}
EXCUTE_ST
    freopen("reverse.in","r",stdin);
    freopen("reverse.out","w",stdout);
    n = read();
    k = read();
    m = read();
    s = read();
    for(int i = 1;i <= m;i++){
        int x;
        x = read();
        est[x] = 1;
    }
    for(int i = 1;i <= n;i++){
        ans[i] = -1;
    }
    for(int i = 1;i <= n;i++){
        if(est[i] || i == s)continue;
        if(i%2 == 0){
            set2.insert(i);
        }else{
            set1.insert(i);
        }
    }
    ans[s] = 0;
    q.push(s);
    while(!q.empty()){
```

```
        const int u = q.front();
        q.pop();
        int l = max(1,u - k + 1),r = min(n,u + k - 1);
        l = l + (l + k - 1) - u,r = r + (r - k + 1) - u;
        set<int> &st = (l%2 == 0)?set2:set1;
        for(auto i = st.lower_bound(l);i != st.end();){
            if(*i > r)break;
            ans[*i] = ans[u] + 1;
            q.push(*i);
            st.erase(i++);
        }
    }
    for(int i = 1;i <= n;i++){
        printf("%d ",ans[i]);
    }
FILE_ED_And_excute
```

# D2T2 - silhouette.cpp

一眼组合数学，做了一下感觉不太可做的亚子，于是写了个暴力搜索，不知道为什么没得到期望分……(T了)

正解就是组合数学，有理解难度但是可以啃一下，绝赞理解中……

# D2T3 - seat.cpp

概率 + DP …… 很不可做。

骗了个 $n = 1$ 和 $n = 2$ 的分就跑了。（解释下，这两道题不是总结很水，而是的确正解还没改出来，然后暴力也没什么含金量）

# D2T4 - ancient.cpp

一眼数论，鉴定为不会做。用杨辉三角求 $C$ 骗了一些分（不知道为什么还是实际分数比起预期分要低）。

正解是数论大礼包——中国剩余定理 + 卢卡斯定理

首先我们把题目转换为：求 $G^{\Sigma d|nC_d^n} \ mod \ 999911659$，使用欧拉定理的推论转化为：$G^{\Sigma d|nC_d^n \ mod \ 999911658} \ mod \ 999911659$，就算咱数论菜成这样也知道这个 $G^n$ 是方便用 `ksm` 求的，那么关键就在上面这一坨。我们放大来看：

$$\Sigma d|nC_d^n \ mod \ 999911658$$

这个 $d|n$ 显然可以枚举，关键就在于求这个 $C_d^n$，如果知道一些数论知识不难想到卢卡斯定理（但是我的数论真的不行，完全没有想到ww），然而这个模数太大依然会炸。然后神仙们就可以想到分解掉这个模数再合并（咱绝对想不到这个）

$999911658 = 2 \times 3 \times 4679 \times 35617$，分解出来以后就可以用卢卡斯来求 $C$ 了，最后用中国剩余定理来解下面这个同余方程组：

$$\begin{cases} x \equiv a_1 (mod \ 2) \\ x \equiv a_2 (mod \ 3) \\ x \equiv a_3 (mod \ 4679) \\ x \equiv a_4 (mod \ 35617) \end{cases}$$

得到 $x$ 以后，用`ksm`求出最终答案 $G^x$。

## 代码

```
#include<bits/stdc++.h>
#define FILE_ST using namespace std;namespace Mi_Shatas_Lilion{
#define EXCUTE_ST int main(){
#define FILE_ED_And_excute return 0;}}int main(){return
Mi_Shatas_Lilion::main();}
#define ll long long
FILE_ST
ll n,g;
const ll MAXJ = 40006;
const ll MOD = 999911659;
const ll pr[14] = {0,2,3,4679,35617};
ll jc[MAXJ],a[14],ans;
void build(ll p){
    jc[0] = 1,jc[1] = 1;
    for(int i = 2;i <= p;i++)jc[i] = (jc[i-1] * i) % p;
    return ;
}
ll ksm(ll a,ll b,ll p){
    ll ret = 1;
    a %= p;
    while(b){
        if(b & 1 == 1)ret = (ret * a) % p;
        a = (a * a) % p;
        b >>= 1;
    }
    return ret;
}
ll C(ll x,ll y,ll p){
    if(x < y)return 0;
    return jc[x] * ksm((jc[y] * jc[x-y])%p,p-2,p) % p;
}
ll lucas(ll x,ll y,ll p){
    if(x < y)return 0;
    if(x == 0)return 1;
    return lucas(x/p,y/p,p) * C(x%p,y%p,p) % p;
}
void crt(){
    for(int i = 1;i <= 4;i++){
        const int m = (MOD-1) / pr[i];
        ans += ((a[i] * m)%(MOD-1)) * ksm(m,pr[i]-2,pr[i]) % (MOD-1);
    }
    return ;
}
EXCUTE_ST
    freopen("ancient.in","r",stdin);
    freopen("ancient.out","w",stdout);
    cin>>n>>g;
    for(int i = 1;i <= 4;i++){
        build(pr[i]);
        for(ll j = 1;j * j <= n;j++){
            if(n % j != 0)continue;
```

```
            a[i] = (a[i] + lucas(n,j,pr[i])) % pr[i];
            if(j*j != n){
                a[i] = (a[i] + lucas(n,n/j,pr[i])) % pr[i];
            }
        }
    }
    crt();
    cout<<ksm(g,ans,MOD);
FILE_ED_And_excute
```

# 总结

感觉这周的考试状态真的不是很好，有以下几点原因。

1. 该骗的分没有骗全，像 $substr.cpp$ 的暴力应该先写好放着再去写正解。
2. 薄弱板块太过于突出：点名指出**数论**
3. 思维太容易卡在某一处：建立了错误的费用流模型后没有想到换模型而是魔改费用流。
4. DP能力还是欠缺，最后需要补一下

# 优良传统

圆焰 += 2　(哈图里找不到图力...本来想放哈图里的安岛的,然而过于直球咱不想社死XD) (在哈图里看到了花吻...这真的是我可以看的吗)

(草，DJC把安岛放上去了ww圆了我的梦，于是我也放一张安岛)(另，咱的代码文件的格式都是 $11.15\_AdachiandShimamura$ 这样的)


1
Adachi
and
Shimamura