1. 人机识别

原题是CF1368B，评分 1500，~~我真不信这能有人签到失败~~！

其实只凭直觉也知道，答案字符串一定是 iamhuman 每个字符重复若干次的结果。如果要证明，那就按照如下顺序依次说明结论：

- i,h,u,n 的出现 **构成连续段**。以 h 为例，每个 h 的贡献一定是 $($ 前面 iam 的出现次数 $)\times($ 后面 uman 的出现次数 $)$，由于这个系数跟 h 无关，所以总是要把所有 h 都放在系数最大的那个位置。
- i,h 之间 **只有** a,m **且** a **在前**。这是显而易见的，因为任意 iamhuman 出现都需要利用这两段仅有的 i 和 h，二者之间的其它字符都是无用的。同理，u,n 之间也是如此。

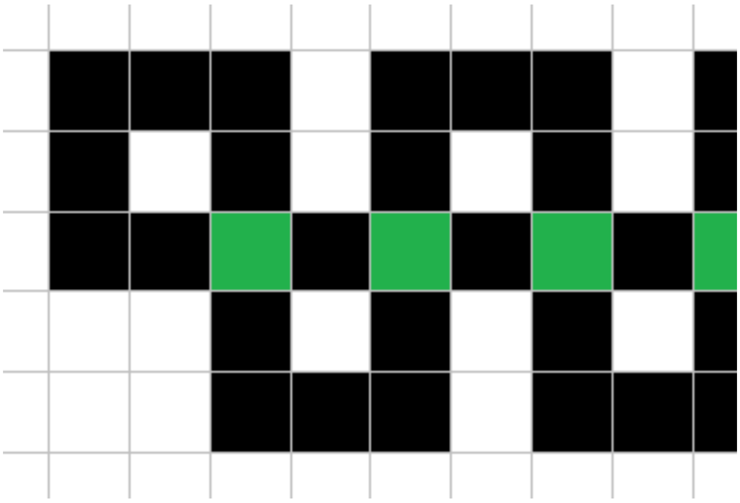
于是就证毕了。如果每个字符的重复次数是 n_i，那么出现次数是 $\prod n_i$，由 $a-b\geqslant 2\;\Rightarrow\;(a{-}1)(b{+}1)=ab{+}(a{-}b{-}1)>ab$ 可知 $|n_i-n_j|\leqslant 1$，于是二分 $\sum n_i$ 后可直接算出每个 n_i 。时间复杂度并不重要 😄

> 补记：如果是任意字符串，这个结论并不成立，比如题面中给出的 abcbc 。对于一般字符串，有通用解法吗？

2. 字符画

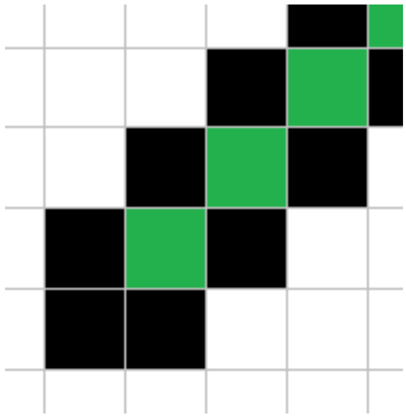
原题是CF1368C，评分 1500 。我略微做了一些改动，却也仍然是最简单之一。

或许选手会得到这样的简单构造方法（~~虽然看上去比其他得分更高的构造方法都难~~）：



其中绿色是费心思的 `#`，而黑色格子是普通的 `#` 。这是 $k=7n+8$ 的，将会得到 35pts 。

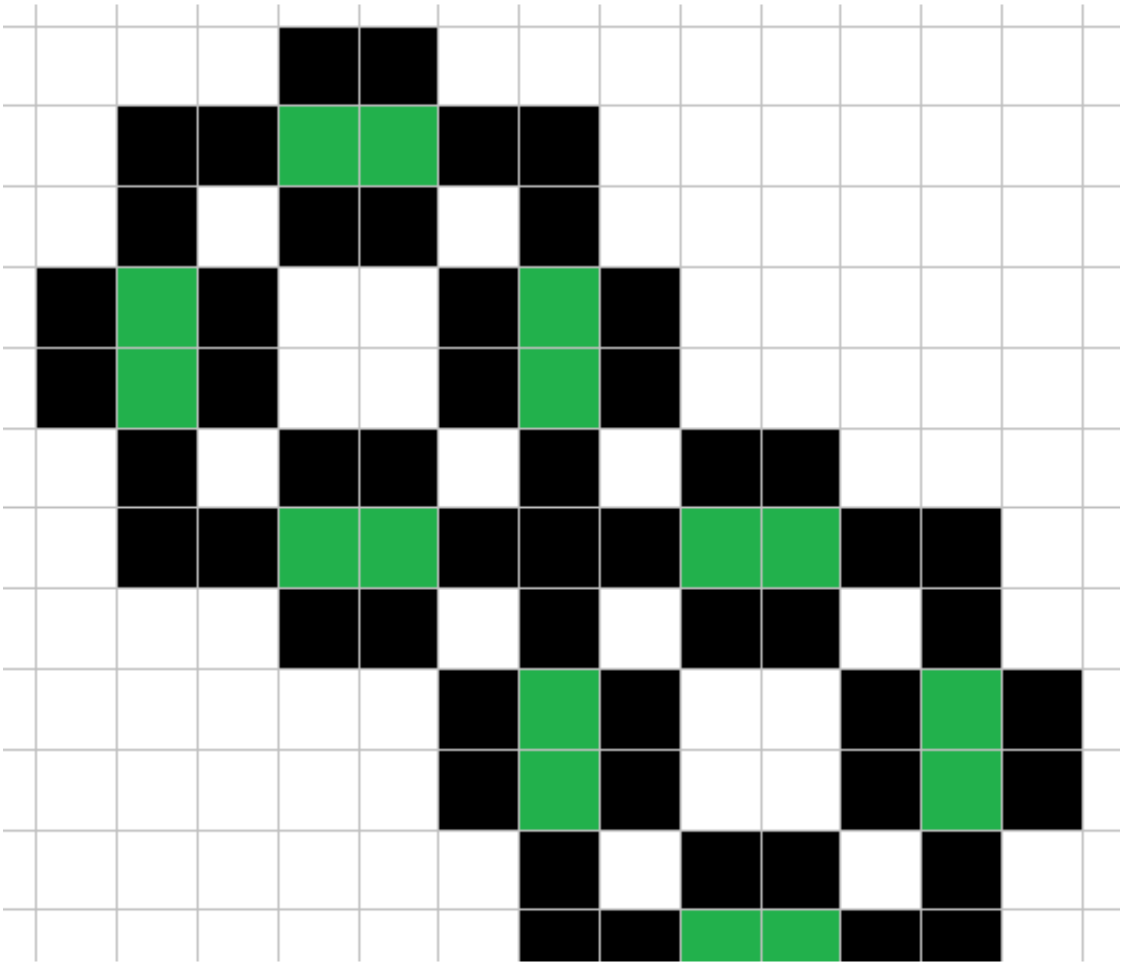
其实最容易想到的应该是这种：



这是 $k=3n+4$ 的，空间利用率极高。这将会得到 80pts 的高分！

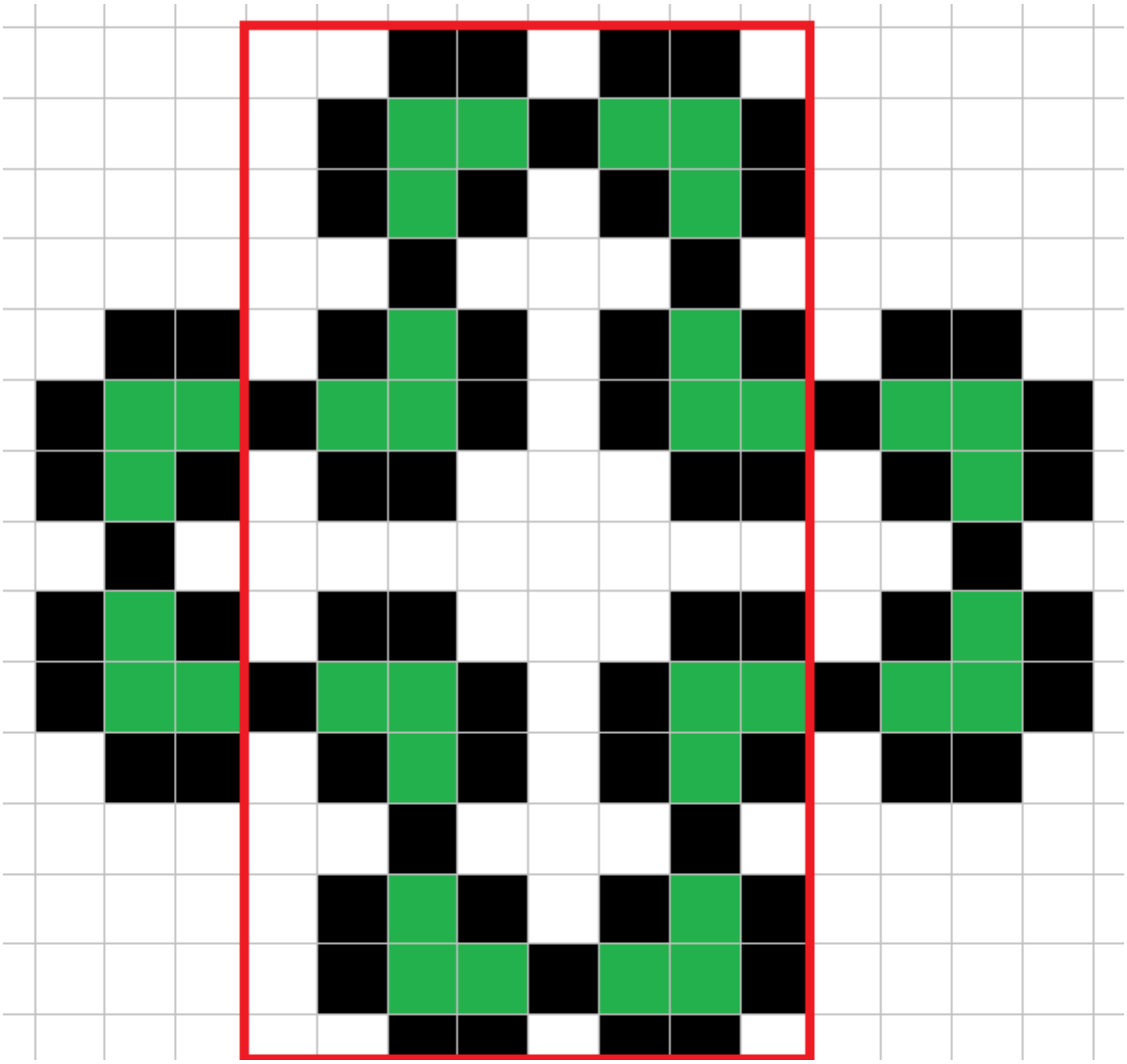
原本我想要证明 $k=3n+4$ 就是下界了。结果被 OneInDark 给 hack 了。考虑到底怎么才能搞出一些 k 更小的方法呢？

考虑绿色格子（费心思的 `#` 号）的四连通大小。如果为 1，那么上面的“斜条”约莫是最好；如果为 2，大概会是



非常不简洁，而且 $k=\frac{9}{2}n$ 较劣，而且只在 $8\mid n$ 时比较好用。不难发现 $n=200$ 就是这一档分，估计没人会去写……

那么连通块的大小扩大到 3 呢？你会惊讶地发现：



长得像一朵花。仔细地数一数，会发现这是 $k=3n$ 的方法。将红框内的部分复制若干次，则可构造出 $n=3(4+8t)\ (t\in\mathbb{N})$ 的方案。不难发现 $t=12$ 时 $n=300$，这就是正解。代码实现可参考 `std` 。

> 补记：连通块大小为 4 则构成正方形，嗝屁（无法扩展）。所以 $k=3n$ 是否为下界?

3. 能量

原题是：hdu 6757

算法 1：

暴力枚举。

期望得分：10 分。

算法 2：

贪心排序策略：$a_i \le b_i$ 一定在 $a_i > b_i$ 之前，$a_i \le b_i$ 内部按 a_i 从小到大，$a_i > b_i$ 内部按 b_i 从大到小。

二进制枚举一下。

期望得分：25 分。

算法 3：

记 $f_{i,j}$ 表示后 i 个选 j 个的最小初始能源，可以做到 $O(n^2)$ 的 dp。

期望得分：40 分。

算法 4：

只有 $a_i \le b_i$，显然取排序后前 k 个最优。

期望得分：15 分。结合算法 3 期望得分 55 分。

算法 5：

将 $a_i \le b_i$ 与 $a_i > b_i$ 分开算，$a_i \le b_i$ 按算法 4 贪心，$a_i > b_i$ 按算法 3 进行 dp。

设 g_i 表示贪心前 i 个最小初始能源，h_i 表示贪心前 i 个的增量，显然有 $g_i \le g_{i+1}, h_i \le h_{i+1}$。

考虑如何合并答案：$ans_k = \min_{i+j=k}\{\max\{g_i, f_{p,j} - h_i\}\}$。可以考虑枚举 g_i，记 $last$ 为上次 ans 最后一次被更新的位置。

先算 $f_{p,j} - h_i \le g_i$ 的最大 j，则 $ans_{last+1...i+j} = g_i$ 并更新 $last$；再算 $f_{p,j} - h_i \le g_{i+1}$ 的最大 j，则 $ans_{last+1...i+j} = f_{p,j} - h_i$。

期望得分：40 分。结合算法 3 期望得分 80。

~~我本以为算法 4 和 5 设计得很好，直到我生成大样例的时候才发现纯随机数据答案全部一样。~~

算法 6：

只考虑 $a_i > b_i$，如何优化 dp。首先显然 $f_{i,j-1} \le f_{i,j}$ 且 $f_{i,j} \le f_{i-1,j}$。

考虑 $\max\{f_{i-1,j-1} + a_i - b_i, a_i\}$ 的取值，它决定于 $f_{i-1,j-1}$ 与 b_i 的大小关系。由于 $f_{i,j-1} \le f_{i,j}$，所以存在 j_0 满足 $j \le j_0$ 时 $f_{i-1,j-1} \le b_i$；$j > j_0$ 时 $f_{i-1,j-1} > b_i$。

注意到 $f_{i,j} \le f_{i-1,j}$ 且 $b_i \le b_{i+1}$（一开始的贪心性质，注意我们是从后往前 dp），则 j_0 随 i 增大而增大。

> 对于 $j < j_0$，有 $f_{i-1,j-1} \le b_i$ 且 $f_{i-1,j} \le b_i < a_i$，故 $f_{i,j} = f_{i-1,j}$。
>
> 对于 $j = j_0$，$f_{i,j} = \min\{f_{i-1,j}, a_i\}$。
>
> 对于 $j > j_0$，$f_{i,j} = \min\{f_{i-1,j}, f_{i-1,j-1} + a_i - b_i\}$。它的值取决于 $f_{i-1,j} - f_{i-1,j-1}$ 与 $a_i - b_i$ 的大小关系。

考虑加入虚点 $f'_{i-1,j_0-1} = b_i$ 来统一 $j \ge j_0$ 的形式，得到 $f_{i,j_0} = \min\{f_{i-1,j_0}, f'_{i-1,j_0-1} + a_i - b_i\}$。因此不难想到当 $j \ge j_0$ 时 f_i 应该是凸的，即 $f_{i,j} - f_{i,j-1} \le f_{i,j+1} - f_{i,j}$。

考虑 $f_{i,j}=f_{i-1,j-1}+a_i-b_i$ 的含义：即点 $(j-1,f_{i-1,j-1})$ 平移到 $(j,f_{i-1,j-1}+a_i-b_i)$。可以看作在凸函数内部的合适位置（即保持斜率单调的位置）加入向量 $(1,a_i-b_i)$，因此直接优先队列维护即可。

为了正确维护与虚点之间的斜率，你可能要先把优先队列头弹出来改正确后再放回去。具体可以参考标程实现。

结合一下算法 5 的贪心 + 合并，期望得分：100 分。

4. 最小生成树

线段树分治加 link-cut-tree 是我觉得最垃圾的做法，毫无美感可言，所以也没给什么分。

首先考虑如果第二类边是链怎么做?直接对链建立线段树，对于每个区间 $[l,r]$，我们只需要维

护 $t[0/1][0/1]$ 表示 l 的连通块是否与 0 连通；r 的连通块是否与 0 连通。

在合并时，需要处理中间两个连通块。如果都不与 0 连通，那么不合法;如果都与 0 连通，不需要花费代价就可以合并;如果只有一边与 0 连通，那么需要花费中间那条二类边的代价。

进一步解释上面的方法，考虑 l,r 的连通块具体是什么样子不需要关心，只需要关心与 0 的连通 性就可以支持合并，因为中间那条边的权值都是一样的。另外对于叶子节点的初始化，只有 $t[1][1]$ 需要设置为 a_i，其他都设置成 0 就行了，正确性不难理解。

推广到普遍的情况，考虑找出第二类边的**等效链**。我们对第二类边单独跑 kruskal，对于每个 连通块都维护其对应的等效链，合并两个连通块的时候，我们也合并对应的等效链，直接把两条链的端点用这条边接起来就行了。

这样做为什么是对的呢?考虑出错的情形是:这条边原来连接 (u,v)，但在等效链上连接了 (x,y)，如果 $(u,x)/(v,y)$不在同一个连通块中就可能出错。但是考虑第一类边时，如果这条二类边起作用，那么根据 kruskal 的过程，**这条边的作用环境仍然是不变的**(也就是多考虑了一些边，这条边起作用时必定有 (u,x)和 (v,y)都连通的性质)

那么跑完 kruskal 之后直接上线段树维护，时间复杂度 $O(nlogn)$。

```
#include <cstdio>
#include <vector>
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;
const int M = 300005;
const int inf = 0x3f3f3f3f;
#define ll long long
#define pb push_back
int read() {
    int x=0,f=1;char c;
    while((c=getchar())<'0' || c>'9') {if(c=='-') f=-1;}
    while(c>='0' && c<='9') {x=(x<<3)+(x<<1)+(c^48);c=getchar();}
    return x*f;
}
void write(ll x)
{
    if(x>=10) write(x/10);
    putchar(x%10+'0');
```

```cpp
}
int n,m,q,a[M],b[M],fa[M],p[M],id[M];
vector<int> s[M],w[M];ll t[M<<2][2][2];
struct node{int u,v,c;}e[M];
int find(int x)
{
    if(x!=fa[x]) fa[x]=find(fa[x]);
    return fa[x];
}
void merge(int u,int v,int c)
{
    u=find(u);v=find(v);if(u==v) return ;
    if(s[u].size()<s[v].size()) swap(u,v);
    fa[v]=u;for(int x:s[v]) s[u].pb(x);
    w[u].pb(c);for(int x:w[v]) w[u].pb(x);
}
void up(int i,int zxy)
{
    int l=i<<1,r=i<<1|1;
    memset(t[i],0x3f,sizeof t[i]);
    for(int a=0;a<2;a++) for(int b=0;b<2;b++)
        for(int c=0;c<2;c++) if(b|c) for(int d=0;d<2;d++)
            t[i][a][d]=min(t[i][a][d],t[l][a][b]
            +t[r][c][d]+((b&c)?0:zxy));
}
void build(int i,int l,int r){
if(l==r)
    {
        for(int j=0;j<2;j++)
            for(int k=0;k<2;k++)
                t[i][j][k]=(j&k)?a[p[l]]:0;
return ; }
    int mid=(l+r)>>1;
    build(i<<1,l,mid);
    build(i<<1|1,mid+1,r);
    up(i,b[mid]);
}
void ins(int i,int l,int r,int x)
{
if(l==r) {
        for(int j=0;j<2;j++)
            for(int k=0;k<2;k++)
                t[i][j][k]=(j&k)?a[p[l]]:0;
        return ;
    }
    int mid=(l+r)>>1;
    if(mid>=x) ins(i<<1,l,mid,x);
    else ins(i<<1|1,mid+1,r,x);
    up(i,b[mid]);
```

```
}
signed main()
{
    freopen("mst.in","r",stdin);
    freopen("mst.out","w",stdout);
    n=read();m=read();
    for(int i=1;i<=n;i++)
        a[i]=read(),fa[i]=i,s[i].pb(i);
    for(int i=1;i<=m;i++)
        e[i].u=read(),e[i].v=read(),e[i].c=read();
    sort(e+1,e+1+m,[&](node a,node b){return a.c<b.c;});
    for(int i=1;i<=m;i++)
        merge(e[i].u,e[i].v,e[i].c);
    for(int i=1;i<n;i++)
        merge(i,i+1,inf);
    int rt=find(1),cnt=0;
    for(int x:s[rt]) p[++cnt]=x;
    cnt=0;for(int x:w[rt]) b[++cnt]=x;
    for(int i=1;i<=n;i++) id[p[i]]=i;
    build(1,1,n);q=read();
    while(q--)
    {
} }
```