# 第17届北京理工大学“连山科技”程序设计竞赛(网络选拔赛) - 题解(部分)

## D

偏差认知、碎片整理、扩容直接模拟就行。

主要是模拟生成和激发，以及漆黑的全局加操作。

用一个队列来维护球的数值，生成时入队，激发时出队。注意判断有没有超过球数上限，超了需要激发并出队一下。

用一个数字来记录全局加的值，每次漆黑的时候，该数值+x，队尾打个标记。

某个球出队后需要将全局加的值减去这个标记值。

这样就实现了将当前队列全局加的操作。

答案是取模的，但是中间计算过程需要longlong，比如集中需要用longlong存(因为取模了不知道和0谁大谁小)。

## E

首先一个暴力的想法是每次扫一遍字符串，然后将这两种字符分别替换即可，复杂度$O(nm)$，显然不能通过。

我们考虑这个交换$(x,y)$的本质，实际上是将串里的所有$x$视为$y$，所有$y$视为$x$。如果我们将“视为”记作$x \to x$的形式，那么这一操作会将【$x \to x,y \to y$】变为【$y \to x,x \to y$】，但是要注意的是，这里并不是【交换$x$和$y$所视为的符号】，而是【交换视为$x$和$y$的符号】，请注意区分。

不难发现这个操作具有传递性。即对于$(x,y),(y,z)$，本质上是【$x \to x,y \to y,z \to z$】【$y \to x,x \to y,z \to z$】【$y \to x,z \to y,x \to z$】这样变化。

于是我们令$ch[i]$表示符号$i$当前视为$ch[i]$，$pos[i]$表示视为$i$当前的是符号$pos[i]$，那么一次操作$(x,y)$，只需要依次执行下面两步即可：

- swap($ch[pos[x]],ch[pos[y]]$)
- swap($pos[x],pos[y]$)

最后，我们可以利用$ch$数组来得到最终的结果，即$S[i] \to ch[S[i]]$

复杂度$O(n+m)$

当然，由于Link的要求，我们放宽了时限，所以可以不用记这个$pos$数组，我们可以直接通过遍历一遍$ch$数组，得到等价于$pos$的信息。

```
for(int i=33;i<127;++i){
    if(ch[i]==x) ch[i]=y;
    else if(ch[i]==y) ch[i]=x;
}
```

复杂度$O(m \cdot C)$，其中$C$是字符范围

# F

注意到最优策略与期望值仅与当前剩余的字数以及当前保存阈值所处的范围有关，假设当前剩余$i$个字，保存阈值在$l \sim r$范围内，若我们决定下一步输入$x$个字后点击保存键，可以分类讨论接下来会发生的情况：

- $x \le l$，必定能够保存成功，当剩余的词数足够大的时候必然尽可能选 $l$ 更优；
- $l < x \le r$，其中有 $\frac{x-l}{r-l+1}$ 的概率保存阈值小于 $x$，此时保存失败，但我们得到了额外信息，保存阈值的上界 $r$ 变为 $x-1$，另外 $\frac{r-x+1}{r-l+1}$ 的概率保存阈值大于等于 $x$，此时保存成功，阈值下界 $l$ 变为 $x$，剩余单词数变为 $i-x$；
- $r < x$，此时必然保存失败，不可能选择这种情况；

我们设剩余词数为 $i$，保存阈值范围为 $l \sim r$ 时的最优期望为$E_{i,l,r}$，根据期望的性质，我们可以得知，在 $l < x \le r$ 时，若我们当前这一步的决策为 $x$，则这种决策的最优期望为 $\frac{x-l}{r-l+1}E_{i,l,x-1} + \frac{r-x+1}{r-l+1}E_{i-x,x,r} + x + t$。根据这个性质，我们可以采用动态规划算法计算最优期望值。具体的，我们仍然使用上方$E$的定义作为dp状态，状态转移方程如下:

$$E_{i,l,r} = \max[E_{i-\min(i,l),l,r} + \min(i,l), \max_{x\in(l,r]}(\frac{x-l}{r-l+1}E_{i,l,x-1} + \frac{r-x+1}{r-l+1}E_{i-x,x,r} + x)] + t$$

为了能够同时比较大小与得到取模的结果，我们需要同时维护浮点的dp值和离散dp值，注意到期望的分母仅在$100$以内，且比较大小时均为同一分母相互比较，故两个不同的dp值的差最小为$0.01$，在这个范围内浮点精度误差不会影响大小的比较，只需要注意两个浮点数相等的判断即可。

直接进行状态转移，总复杂度为$O(nlr^2)$，可以通过此题。

# G 分辨矩阵

## 分析

两个思路，搜索或者状压DP，出题人的做法是状压DP，验题人的做法是搜索。

搜索的话，因为矩阵很小，答案上限也很小（至多 $k=6$），搜一搜剪剪枝就过了。

状压DP的提示还是挺明显的，因为 $T \le 50, k \le 6$。根据时间限制估计，大概是个 $2^{\binom{k}{2}}$ 的状压，不然为什么不出个$k \le 10$ 和 $n, m$ 的范围相同?

对于 $n \times m$ 矩阵的每个位置 $(i,j)$，用一个二进制串状压哪些**矩阵对**在这一位不同，记为 $f(i,j)$，这是一个长为 $\binom{k}{2} \le 15$ 的二进制串，可以用`int`存储。

接下来，用 $g[i]$ 表示当前位置集合中，最少选择几个位置，可以分辨二进制状态为 $i$ 的所有矩阵对。

例如 $g[(7)_{10}] = g[(111)_2]$ 表示要同时区分 $(0,1)$、$(0,2)$、$(1,2)$ 矩阵对需要选取的最少位置个数。

如果要区分所有矩阵对，状态应该是为 $(1\cdots1)_2$，共 $k$ 个 $1$，对应的 $10$ 进制为 $\binom{k}{2}-1$。

最后就剩下一个简单的转移了，逐个插入当前矩阵中的每个位置，用 $f$ 更新 $g$：
$g[i|f(j,k)] \leftarrow g[i]+1$，最终答案为 $g[\binom{k}{2}-1]$。

复杂度 $O(Tnm2^{\binom{k}{2}})$。

## 参考代码

```
#include <bits/stdc++.h>
using namespace std;
void sol() {
  int n, m, k;
```

```cpp
  cin >> n >> m >> k;
  vector<vector<string>> s(k, vector<string>(n));
  vector<vector<int>> d(n, vector<int>(n, 0));
  for (int i = 0; i < k; ++i) {
    for (int j = 0; j < n; ++j) {
      cin >> s[i][j];
    }
  }
  auto get = [&](int a, int b) { return a * k - (a + 1) * a / 2 + b - a - 1; };
  vector<vector<int>> f(n, vector<int>(m, 0));
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
      for (int a = 0; a < k; ++a) {
        for (int b = a + 1; b < k; ++b) {
          if (s[a][i][j] != s[b][i][j]) f[i][j] |= 1 << get(a, b);
        }
      }
    }
  }
  int mask = 1 << (k * (k - 1) / 2);
  vector<int> g(mask, 1e9);
  g[0] = 0;
  for (int j = 0; j < n; ++j) {
    for (int k = 0; k < m; ++k) {
      if (f[j][k] == 0) continue;
      for (int i = mask - 1; i >= 0; --i) {
        if (g[i] + 1 < g[i | f[j][k]]) g[i | f[j][k]] = g[i] + 1;
      }
    }
  }
  cout << g[mask - 1] << endl;
}
int main() {
  int cs;
  cin >> cs;
  while (cs--) sol();
  return 0;
}
```