Hello World ! ---KM二分图带权匹配

/\*

KM 二分图带权匹配 简单题

题意：N个箱子围成一圈，每个箱子里可能头0颗或者若干颗巧克力，总数目不超过N。现要求移动这些巧克力使得每个箱子有不超过1个巧克力，求最少步数。移动一步可以把箱子里的一个巧克力放置到相邻的箱子里面。

思路：对有超过1个巧克力的箱子拆成a[i]-1个点作为X部，对有0个巧克力的箱子作为Y部，KM匹配。

\*/

bool find(int u)

{

visx[u] = when;

for(int v = 0; v < m; v++) if(visy[v] != when) {

int rest = LX[u] + LY[v] - edge[u][v];

if(rest == 0) {

visy[v] = when;

if(link[v] == -1 || find(link[v])) {

link[v] = u;

return true;

}

} else slack[v] = min(slack[v], rest);

}

return false;

}

void KM()

{

memset(LX, 128, sizeof(LX));

memset(LY, 0, sizeof(LY));

memset(link,-1, sizeof(link));

for(int i = 0; i < n; i++) for(int j = 0; j < m; j++) LX[i] = max(LX[i], edge[i][j]);

for(int start = 0; start < n; start ++) {

for(int i = 0; i < m; i++) slack[i] = (INT\_MAX);

while(true) {

when++;

if(find(start)) break;

else {

int d = (INT\_MAX);

for(int i = 0; i < m; i++) if(visy[i] != when) d = min(d, slack[i]);

for(int i = 0; i < n; i++) if(visx[i] == when) LX[i] -= d;

for(int i = 0; i < m; i++)

if(visy[i] == when) LY[i] += d;

else slack[i] -= d;

}

}

}

}

int main()

{

int x;

while(scanf("%d", &N) != EOF) {

n = m = 0;

for(int i = 0; i < N; i++) {

scanf("%d", &x);

if(x == 0) id[0][m++] = i;

if(x > 1) {

while(x > 1) {

x--;

id[1][n++] = i;

}

}

}

for(int i = 0; i < n; i++)

for(int j = 0; j < m; j++) {

edge[i][j] = -dis(id[1][i], id[0][j]);

}

KM();

int ans = 0;

for(int i = 0; i < m; i++) if(link[i] != -1) ans += LX[link[i]] + LY[i];

printf("%d\n", -ans);

}

return 0;

}

/\*

二分图带权匹配 KM 经典题 好题

题意：一个有向图，把该图分成一些环，使得构成这些环的总权值最小。一个环至少两个点，一个点在且仅在一个环中。

思路：神奇的构图...orz...

环的性质其实就是每个点有且仅有一个入度一个出度。拆点并把原边(u,v)变成(u,v')就把该环的性质转化成了二分图匹配...加上权值，就是二分图带权匹配了。

建图虽然用一句话就概括完了，但很妙好不好！所以说，图论的构图真是靠运气成分很大呀。

\*/

//hdu 1853

int main()

{

int m, x, y, z;

while(scanf("%d%d", &n, &m) != EOF) {

init();

for(int i = 0; i < m; i++) {

scanf("%d%d%d", &x, &y, &z);

add\_edge(x, hash(y), -z);

}

bool perfect\_match = KM();

if(!perfect\_match) {

printf("-1\n");

continue;

}

int ans = 0;

for(int i = n+1; i <= n+n; i++) ans += LX[link[i]] + LY[i];

printf("%d\n", -ans);

}

return 0;

}

/\*

KM 二分图带权匹配

题意：两个字符串的similarity定义为：

Similarity(S, T) = sum(Si == Ti) / L

L = Length(S) = Length(T), i = 1, 2,... L, where sum(Si == Ti) indicates the total number of equal labels in corresponding positions.

就是算两个串的最大similarity.

思路：按字母定义点。边权用上式算，其实只要扫一遍即可，在build里面算边权.

\*/

//zoj 3425

void build()

{

memset(dis, 0, sizeof(dis));

for(int i = 0; i < N; i++)

dis[map[0][a[i]]][map[1][b[i]]] ++; //算边权

}

int main()

{

int cases, K, M;

scanf("%d", &cases);

while(cases--) {

scanf("%d%d%d", &N, &K, &M);

memset(boo, false, sizeof(boo));

m = 0;

for(int i = 0; i < N; i++) {

char ch = rdc();

b[i] = ch;

if(!boo[ch]) boo[ch] = true, map[1][ch] = m++;

}

while(M--) {

memset(boo, false, sizeof(boo));

n = 0;

for(int i = 0; i < N; i++) {

char ch = rdc();

a[i] = ch;

if(!boo[ch]) boo[ch] = true, map[0][ch] = n++;

}

build();

KM();

int sum = 0;

for(int i = 0; i < m; i++) if(link[i] != -1) sum += LX[link[i]] + LY[i];

printf("%.4f\n", sum\*1.0/N);

}

}

return 0;

}

/\*

KM 二分图匹配 MCMF DP 搜索 好题

题意：在8\*8棋盘上，给出8个皇后的初始位置，要求移动这8个皇后使得皇后之间互不攻击。皇后移动一步可以沿直线或对角线移动任意个格子，但不能跨过一个原本有皇后的格，即会被有皇后的格挡住。皇后的攻击方式跟移动方式一样的。 问最小移动总步数。

思路：neko最初给我看的时候我第一反应是状压DP，但感觉很复杂，搞不动。

他说只要能证明一个结论，这道题就很简单了。要证：存在一种最终布局，使得最优移动过程中任何有皇后不互挡，也就是说我们可以不用考虑“不能跨皇后”这个限制，直接“穿越”做。

我画了个4\*4的图，然后幸运地得出了证明：假设现在在i点的皇后要到k点去（那k点当然是最终布局的皇后点啦，不然到那里去干嘛），中间有个j（j点有个皇后）隔住了。 那么k与j就在一条直线上，也就是说 j 不是最终皇后点，那么 j 就应该要挪到她相应的点去，等她挪完后就不挡住i 到达 k 点了。

证了上述结论就好办了，用二分图带权匹配KM算法或者最大流最小费用流MCMF都可以做，费用为移动步数。

我是用KM 做的。先预处理出8\*8上所有合理布局，一共92种。然后对于每种合理布局，拿来跟初始布局搞二分图带权匹配。易知这个二分图是完全二分图（表述不太准确，就是X部Y部两两有边）。

这道题是归类在DP上的 orz... 在vj上有人用搜索+dp做...

\*/

//lightoj 1061

#include <limits.h>

#include <stdio.h>

#include <algorithm>

#include <string.h>

using namespace std;

#define N 8

#define ABS(x) ((x) > 0 ? (x) : (-(x)))

//bool flag = false; //for debug

int l[3][N\*2], X[N], map[100][N], a[N][2]; //map[i][x] 存第i种合理布局的x行皇后应该放的列数

int edge[N][N], LX[N], LY[N], link[N], slack[N];

bool visx[N], visy[N];

int Top = 0;

void label(int row, int col)

{

l[0][col] ^= 1;

l[1][row+col] ^= 1;

l[2][row-col+N] ^= 1;

}

bool check(int row, int col)

{

return l[0][col] && l[1][row+col] && l[2][row-col+N];

}

void init(int row)

{

if(row == N) {

for(int i = 0; i < N; i++) map[Top][i] = X[i];

Top++;

return ;

}

for(int i = 0; i < N; i++) {

if(check(row, i)) {

X[row] = i;

label(row, i);

init(row+1);

label(row, i);

}

}

}

inline char rdc() { scanf(" "); return getchar(); }

void read\_board()

{

int k = 0;

for(int i = 0; i < N; i++)

for(int j = 0; j < N; j++)

if(rdc() == 'q') a[k][0] = i, a[k][1] = j, k++;

}

int dis(int x1, int y1, int x2, int y2)

{

if(x1 == x2 && y1 == y2) return 0;

if(x1 == x2 || y1 == y2) return 1;

if(x1 + y1 == x2 + y2 || (x1 - y1) == (x2 - y2)) return 1; //妹的，本来写的是对的，看了neko的，然后改错了=\_=

return 2;

}

void build(int \* map)

{

for(int i = 0; i < N; i++) {

int x = a[i][0], y = a[i][1];

for(int j = 0; j < N; j++) {

edge[i][j] = -dis(x, y, j, map[j]);

}

}

}

bool find(int u)

{

visx[u] = true;

for(int v = 0; v < N; v++) if(!visy[v]) {

int rest = LX[u] + LY[v] - edge[u][v];

if(rest == 0) {

visy[v] = true;

if(link[v] == -1 || find(link[v])) {

link[v] = u;

return true;

}

} else slack[v] = min(slack[v], rest);

}

return false;

}

int KM()

{

memset(LX, 128, sizeof(LX));

memset(LY, 0, sizeof(LY));

memset(link,-1, sizeof(link));

for(int i = 0; i < N; i++) for(int j = 0; j < N; j++) LX[i] = max(LX[i], edge[i][j]);

for(int start = 0; start < N; start ++) {

for(int i = 0; i < N; i++) slack[i] = (INT\_MAX);

while(true) {

memset(visx, false, sizeof(visx));

memset(visy, false, sizeof(visy));

if(find(start)) break;

else {

int d = (INT\_MAX);

for(int i = 0; i < N; i++) if(!visy[i]) d = min(d, slack[i]);

//if(d == (INT\_MAX)) return -1;

for(int i = 0; i < N; i++) if(visx[i]) LX[i] -= d;

for(int i = 0; i < N; i++)

if(visy[i]) LY[i] += d;

else slack[i] -= d;

}

}

}

int sum = 0;

for(int i = 0; i < N; i++) sum += edge[link[i]][i];

return -sum;

}

int main()

{

int cases, Cas = 0;

for(int i = 0; i < 3; i++) for(int j = 0; j < 2\*N; j++) l[i][j] = 1;

init(0);

scanf("%d", &cases);

while(cases--) {

read\_board();

int ans = 17;

for(int i = 0; i < Top; i++) { //枚举每一种最终布局

build(map[i]);

//if(KM() == 6 && ans != 6) {

// for(int j = 0; j < N; j++) printf("(%d,%d) ", a[j][0]+1,a[j][1]+1);

// printf("\n");

// for(int j = 0; j < N; j++) printf("(%d,%d) ", j+1, 1+map[i][j]);

// printf("\n");

//}

ans = min(ans, KM());

}

printf("Case %d: %d\n", ++Cas, ans);

}

return 0;

}