一

1. **const** **int** maxn = 25;
2. **int** G[maxn][maxn];
3. **int** vst[maxn];
4. **int** ans[maxn];
5. **int** t;
6. **int** n;
8. **void** dfs(**int** x, **int** cot){
9. **if**(x == n && cot == 20){
10. printf("%d:  %d ", t++, n);
11. **for**(**int** i = 0; i <= 19; ++i){
12. **if**(i)
13. printf(" ");
14. printf("%d", ans[i]);
15. }
16. printf("\n");
17. **return** ;
18. }
20. **for**(**int** i = 1; i <= 20; ++i){
21. **if**(!vst[i] && G[x][i]){
22. vst[i] = 1;
23. ans[cot] = i;
24. dfs(i, cot+1);
25. vst[i] = 0;
26. }
27. }
28. }

31. **int** main(){
32. **for**(**int** i = 1; i <= 20; ++i){
33. **int** x, y, z;
34. scanf("%d%d%d", &x, &y, &z);
35. G[i][x] = G[i][y] = G[i][z] = 1;
36. }
37. t = 1;
38. **while**(~scanf("%d", &n)){
39. **if**(!n)
40. **return** 0;
41. **for**(**int** i = 1; i <= 20; ++i){
42. vst[i] = 0;
43. ans[i] = 0;
44. }
45. **for**(**int** i = 1; i <= 20; ++i){
46. **if**(G[n][i]){
47. ans[0] = i;
48. vst[i] = 1;
49. dfs(i, 1);
50. vst[i] = 0;
51. }
52. }
53. }
54. **return** 0;
55. }

二

1. **const** **int** INF = 1e8;
2. **const** **int** MAXN = 4e5;
3. **int** n,m;
4. **char** eight\_num[9][10] = {"X12345678","1X2345678","12X345678","123X45678","1234X5678","12345X678",
5. "123456X78","1234567X8","12345678X"};
6. **char** step[] = "dlru";
7. **int** dx[4] = {1,0,0,-1};
8. **int** dy[4] = {0,-1,1,0};
9. **int** fac[9] = {1,1,2,6,24,120,720,5040,40320};
10. **int** vis[9][MAXN];
11. **int** pre[9][MAXN];
12. **struct** node{
13. **int** num[9];//八数码
14. **int** state;//状态（康托展开值）
15. **int** pos;//x的位置
16. };
17. //得到状态
18. **int** Cantor(**int** \*s){
19. **int** sum = 0;
20. **for**(**int** i = 0; i < 9; i++){
21. **int** num = 0;
22. **for**(**int** j = i+1; j < 9; j++){
23. **if**(s[i] > s[j]) num++;
24. }
25. sum += num \* fac[8-i];
26. }
27. **return** sum;
28. }
29. **void** printAns(**int** state,**int** kind){
30. string ans;
31. **while**(state != -1){
32. ans += step[vis[kind][state]];
33. state = pre[kind][state];
34. }
35. printf("%d\n",ans.size()-1);
36. **for**(**int** i = ans.size() - 2; i >= 0; i--){
37. printf("%c",ans[i]);
38. }
39. puts("");
40. }
41. **void** bfs(node x,**int** kind){
42. queue<node> q;
43. q.push(x);
44. vis[kind][x.state] = 1;//这个是初始化的值并不是真正走的值所以上面输出的时候是从倒数第二个开始输出
45. **while**(!q.empty()){
46. node u = q.front();
47. q.pop();
48. //4种交换
49. **int** x = u.pos / 3;
50. **int** y = u.pos % 3;
51. **for**(**int** i = 0; i < 4; i++){
52. **int** xx = x + dx[i];
53. **int** yy = y + dy[i];
54. **if**(xx >= 0 && xx < 3 && yy >= 0 && yy < 3){
55. node v = u;
56. swap(v.num[u.pos],v.num[xx\*3+yy]);
57. v.state = Cantor(v.num);
58. **if**(vis[kind][v.state] == -1){
59. v.pos = xx \* 3 + yy;
60. vis[kind][v.state] = i;//代表通过i方向的移动到达kind这类中的v.state状态
61. pre[kind][v.state] = u.state;//记录这种状态的前一种状态
62. q.push(v);
63. }
64. }
65. }
66. }
67. }
68. **void** init(**char** \*s,**int** kind){
69. node u;
70. **for**(**int** i = 0; i < 9; i++){
71. **if**(s[i] == 'X'){
72. u.num[i] = 0;
73. u.pos = i;
74. }
75. **else** u.num[i] = s[i] - '0';
76. }
77. u.state = Cantor(u.num);
78. bfs(u,kind);
79. }
80. **int** main(){
81. **int** t,cas = 0;
82. **char** s[15],c[15];
83. **int** kind;
84. **int** temp[MAXN];
85. memset(vis,-1,**sizeof**(vis));
86. memset(pre,-1,**sizeof**(pre));
87. **for**(**int** i = 0; i < 9; i++){
88. init(eight\_num[i],i);
89. }
90. scanf("%d",&t);
91. **while**(t--){
92. scanf("%s",s);
93. **int** cnt = 1;
94. //映射初始状态的八数码
95. **for**(**int** i = 0; i < 9; i++){
96. **if**(s[i] == 'X'){
97. c[0] = 0;
98. kind = i;
99. }
100. **else** c[s[i]-'0'] = cnt++;
101. }
102. //映射目标状态八数码
103. scanf("%s",s);
104. **for**(**int** i = 0; i < 9; i++){
105. **if**(s[i] == 'X') temp[i] = c[0];
106. **else** temp[i] = c[s[i]-'0'];
107. }
108. **int** state = Cantor(temp);
109. printf("Case %d: ",++cas);
110. printAns(state,kind);
111. }
112. **return** 0;
113. }

三

1. **const** **int** maxn = 110;
2. **bool** mp[maxn][maxn], Time[maxn][maxn][1010];
3. **bool** vst[maxn][maxn][1010];
4. **int** ans;
5. **int** step[5][2] = {-1,0, 1,0, 0,1, 0,-1,0,0};
6. **int** n, m, k, d;
8. **struct** Point1{
9. **int** x, y, t, v;
10. **int** dir;
11. Point1(**int** xx, **int** yy, **int** tt, **int** vv, **int** direction):x(xx),y(yy),t(tt),v(vv),dir(direction){}
12. Point1(){}
13. };
14. Point1 ta[maxn];
16. **struct** Point2{//角色
17. **int** x, y, t;
18. Point2(**int** xx, **int** yy, **int** tt):x(xx),y(yy),t(tt){}
19. Point2(){}
20. };
22. **bool** check(**int** x, **int** y){   //x y 可以取n m
23. **if**(x < 0 || x > n || y < 0 || y > m || mp[x][y])
24. **return** **false**;
25. **return** **true**;
26. }
28. **void** pre\_set(){
29. **for**(**int** i = 0; i <= n; ++i)
30. **for**(**int** j = 0; j <= m; ++j)
31. **for**(**int** t = 0; t <= d; ++t)
32. Time[i][j][t] = vst[i][j][t] = 0;
33. **for**(**int** i = 0; i<k; i++)    //枚举城堡
34. **for**(**int** j = 0; j<=d; j += ta[i].t)    //模拟一颗子弹
35. **for**(**int** t = 1; ; t++){   //枚举路程
36. **int** x = ta[i].x + step[ta[i].dir][0]\*t;
37. **int** y = ta[i].y + step[ta[i].dir][1]\*t;
38. **if**( !check(x, y) )
39. **break**;
40. **if**(t%ta[i].v==0)    //到达整点时刻，更新hav数组
41. Time[x][y][j+t/ta[i].v] = **true**;
42. }
43. }
45. **void** bfs(){
46. queue<Point2> q;
47. Point2 p = Point2(0, 0, 0);
48. vst[0][0][0] = **true**;
49. q.push(p);
50. **while**(!q.empty()){
51. p = q.front();
52. q.pop();
53. **if**(p.t > d){  //累死了
54. ans = -1;
55. **return** ;
56. }
57. **if**(p.x == n && p.y == m){    //顺利回营
58. ans = p.t;
59. **return** ;
60. }
61. **for**(**int** i = 0; i<5; i++){
62. **int** x = p.x + step[i][0];
63. **int** y = p.y + step[i][1];
64. **int** t = p.t + 1;
65. **if**(check(x, y) && !Time[x][y][t] && !vst[x][y][t]){
66. vst[x][y][t] = 1;
67. q.push(Point2(x, y, t));
68. }
69. }
70. }
71. }
73. **int** main(){
74. **while**(scanf("%d%d%d%d",&n,&m,&k,&d)!=EOF){
75. **for**(**int** i = 0; i <= n; ++i)
76. **for**(**int** j = 0; j <= m; ++j)
77. mp[i][j] = 0;
78. **char** ch[3];
79. ans = -1;
80. **for**(**int** i = 0; i<k; i++){
81. **int** x, y, z, w, dir;
82. scanf("%s%d%d%d%d", ch, &z, &w, &x, &y);
83. **if**(ch[0] == 'N')      dir = 0;
84. **else** **if**(ch[0] == 'S') dir = 1;
85. **else** **if**(ch[0] == 'E') dir = 2;
86. **else** **if**(ch[0] == 'W') dir = 3;
87. ta[i] = Point1(x, y , z, w, dir);
88. mp[x][y] = 1;
89. }
90. pre\_set();
91. bfs();
92. **if**(ans==-1)
93. puts("Bad luck!");
94. **else**
95. printf("%d\n", ans);
96. }
97. }

四

1. **const** **int** maxn = 10;
2. **const** **int** inf = 0x3f3f3f3f;
3. map<**char**, **int**> Hash;
4. **char** s[maxn][maxn];
5. **int** len[maxn], match[maxn];
6. **int** n, MAX, pd;
7. **int** cost[4], tp[4];
8. **char** ch[] = {'A', 'G', 'C', 'T'};
10. //统计每个序列里面四种序列值，并求得每种序列值的最长长度。将四种序列值加起来就是最长待匹配长度。
11. **int** calc(){
12. **for**(**int** i = 0; i < 4; ++i)//4不是maxn
13. cost[i] = tp[i] = 0;
14. memset(cost, 0, **sizeof**(cost));
15. memset(tp, 0, **sizeof**(tp));
16. **for**(**int** i = 0; i < n; ++i){
17. **for**(**int** j = match[i]; j < len[i]; ++j)//每种序列值的长度
18. tp[ Hash[ s[i][j] ] ]++;
19. **for**(**int** j = 0; j < 4; ++j){//每种序列值的最长长度
20. cost[j] = max(cost[j], tp[j]);
21. tp[j] = 0;
22. }
23. }
24. **int** sum = 0;
25. **for**(**int** i = 0; i < 4; ++i)
26. sum += cost[i];
27. **return** sum;
28. }
30. **int** dfs(**int** cot){
31. **int** deep = calc();
32. **if**(!deep){
33. printf("%d\n", MAX);
34. **return** 1;
35. }
36. **if**(deep + cot > MAX){
37. pd = min(deep+cot, pd);
38. **return** 0;
39. }
40. **int** old[maxn];
41. memcpy(old, match, **sizeof**(old));
42. **for**(**int** i = 0; i < 4; ++i){
43. **char** c = ch[i];
44. **int** flag = 0;
45. **for**(**int** j = 0; j < n; ++j){
46. **if**(match[j] < len[j] && s[j][match[j]] == c){
47. flag = 1;
48. ++match[j];
49. }
50. }
51. **if**(flag && dfs(cot + 1))
52. **return** 1;
53. memcpy(match, old, **sizeof**(old));
54. }
55. }
57. **int** main(){
58. **for**(**int** i = 0; i < 4; ++i)
59. Hash[ch[i]] = i;
60. **int** t;
61. scanf("%d", &t);
62. **while**(t--){
63. MAX = 0;
64. memset(match, 0, **sizeof**(match));
65. scanf("%d", &n);
66. **for**(**int** i = 0; i < n; ++i){
67. scanf("%s", s[i]);
68. len[i] = strlen(s[i]);
69. MAX = max(MAX, len[i]);
70. }
71. **while**(1){
72. pd = inf;
73. **if**(dfs(0))
74. **break**;
75. MAX = pd;
76. }
77. }
78. **return** 0;
79. }

五

1. **typedef** pair<**int**, **int**> P;
2. #define N 55
3. #define M 1001
4. **const** **int** INF = 0x3f3f3f3f;
5. #define fi first
6. #define se second
7. //
8. //         1  2  3
9. //         4  5  6
10. //         7  8  9
11. //10 11 12 13 14 15 16 17 18 19 20 21
12. //22 23 24 25 26 27 28 29 30 31 32 33
13. //34 35 36 37 38 39 40 41 42 43 44 45
14. //         46 47 48
15. //         49 50 51
16. //         52 53 54
17. //
18. **int** cent[7] = {23, 26, 29, 32, 5, 50 };
19. **int** face[7][10] = {
20. {10,11,12,22,23,24,34,35,36},
21. {13,14,15,25,26,27,37,38,39},
22. {16,17,18,28,29,30,40,41,42},
23. {19,20,21,31,32,33,43,44,45},
24. {1,2,3,4,5,6,7,8,9},
25. {46,47,48,49,50,51,52,53,54}
26. };
28. **int** change[12][20] = {
29. {1,4,7,13,25,37,46,49,52,21,33,45,10,11,12,24,36,35,34,22},
30. {45,33,21,1,4,7,13,25,37,52,49,46,34,22,10,11,12,24,36,35},
31. {7,8,9,16,28,40,48,47,46,36,24,12,13,14,15,27,39,38,37,25},
32. {36,24,12,7,8,9,16,28,40,48,47,46,37,25,13,14,15,27,39,38},
33. {9,6,3,19,31,43,54,51,48,39,27,15,16,17,18,30,42,41,40,28},
34. {39,27,15,9,6,3,19,31,43,54,51,48,40,28,16,17,18,30,42,41},
35. {42,30,18,3,2,1,10,22,34,52,53,54,19,20,21,33,45,44,43,31},
36. {52,53,54,42,30,18,3,2,1,10,22,34,43,31,19,20,21,33,45,44},
37. {15,14,13,12,11,10,21,20,19,18,17,16,1,2,3,6,9,8,7,4},
38. {18,17,16,15,14,13,12,11,10,21,20,19,7,4,1,2,3,6,9,8},
39. {37,38,39,40,41,42,43,44,45,34,35,36,46,47,48,51,54,53,52,49},
40. {34,35,36,37,38,39,40,41,42,43,44,45,52,49,46,47,48,51,54,53}
41. };
43. **char** s[N];
44. **int** t, res;
45. **int** a[10], b[10];
47. **char** getch(){
48. **char** c; **while**((c = getchar()) == ' ' || c == '\n');
49. **return** c;
50. }
52. **int** geth(**char** \* s){
53. **int** mx = 0;
54. **for**(**int** i = 0; i < 6; ++i)
55. **for**(**int** j = 0; j < 9; ++j)
56. **if**(s[face[i][j]] != s[cent[i]])
57. mx++;
58. **return** (mx + 11) / 12;
59. }
61. **bool** dfs(**int** deep){
62. **int** t = geth(s);
63. **if**(t + deep > res) **return** **false**;
64. **if**(!t) **return** **true**;
66. **char** tmp[N];
67. memcpy(tmp, s, **sizeof** s);
68. **for**(**int** i = 0; i < 12; ++i){
69. **for**(**int** j = 0; j < 20; ++j)
70. s[change[i][j]] = tmp[change[i^1][j]];
72. a[deep] = i / 2;
73. b[deep] = i & 1?-1:1;
75. **if**(dfs(deep + 1)) **return** **true**;
76. memcpy(s, tmp, **sizeof** tmp);
77. }
78. **return** **false**;
79. }
81. **void** ida(){
82. res = geth(s);
84. **if**(!res) {printf("0\n"); **return**;}
85. res = 1;
86. **while**(res <= 5){
87. **if**(dfs(0)){
88. printf("%d\n", res);
89. **for**(**int** i = 0; i < res; ++i)
90. printf("%d %d\n", a[i], b[i]);
91. **return**;
92. }
93. res++;
94. }
95. printf("-1\n");
96. }
98. **int** main(){
99. scanf("%d", &t);
100. **while**(t--){
101. **for**(**int** i = 1; i < N; ++i)
102. s[i] = getch();
103. ida();
104. }
106. **return** 0;
107. }

六

1. **typedef** pair<**int**, **int**> P;
2. #define N 55
3. #define M 1001
4. **const** **int** INF = 0x3f3f3f3f;
5. #define fi first
6. #define se second
7. //
8. //         1  2  3
9. //         4  5  6
10. //         7  8  9
11. //10 11 12 13 14 15 16 17 18 19 20 21
12. //22 23 24 25 26 27 28 29 30 31 32 33
13. //34 35 36 37 38 39 40 41 42 43 44 45
14. //         46 47 48
15. //         49 50 51
16. //         52 53 54
17. //
18. **int** cent[7] = {23, 26, 29, 32, 5, 50 };
19. **int** face[7][10] = {
20. {10,11,12,22,23,24,34,35,36},
21. {13,14,15,25,26,27,37,38,39},
22. {16,17,18,28,29,30,40,41,42},
23. {19,20,21,31,32,33,43,44,45},
24. {1,2,3,4,5,6,7,8,9},
25. {46,47,48,49,50,51,52,53,54}
26. };
28. **int** change[12][20] = {
29. {1,4,7,13,25,37,46,49,52,21,33,45,10,11,12,24,36,35,34,22},
30. {45,33,21,1,4,7,13,25,37,52,49,46,34,22,10,11,12,24,36,35},
31. {7,8,9,16,28,40,48,47,46,36,24,12,13,14,15,27,39,38,37,25},
32. {36,24,12,7,8,9,16,28,40,48,47,46,37,25,13,14,15,27,39,38},
33. {9,6,3,19,31,43,54,51,48,39,27,15,16,17,18,30,42,41,40,28},
34. {39,27,15,9,6,3,19,31,43,54,51,48,40,28,16,17,18,30,42,41},
35. {42,30,18,3,2,1,10,22,34,52,53,54,19,20,21,33,45,44,43,31},
36. {52,53,54,42,30,18,3,2,1,10,22,34,43,31,19,20,21,33,45,44},
37. {15,14,13,12,11,10,21,20,19,18,17,16,1,2,3,6,9,8,7,4},
38. {18,17,16,15,14,13,12,11,10,21,20,19,7,4,1,2,3,6,9,8},
39. {37,38,39,40,41,42,43,44,45,34,35,36,46,47,48,51,54,53,52,49},
40. {34,35,36,37,38,39,40,41,42,43,44,45,52,49,46,47,48,51,54,53}
41. };
43. **char** s[N];
44. **int** t, res;
45. **int** a[10], b[10];
47. **char** getch(){
48. **char** c; **while**((c = getchar()) == ' ' || c == '\n');
49. **return** c;
50. }
52. **int** geth(**char** \* s){
53. **int** mx = 0;
54. **for**(**int** i = 0; i < 6; ++i)
55. **for**(**int** j = 0; j < 9; ++j)
56. **if**(s[face[i][j]] != s[cent[i]])
57. mx++;
58. **return** (mx + 11) / 12;
59. }
61. **bool** dfs(**int** deep){
62. **int** t = geth(s);
63. **if**(t + deep > res) **return** **false**;
64. **if**(!t) **return** **true**;
66. **char** tmp[N];
67. memcpy(tmp, s, **sizeof** s);
68. **for**(**int** i = 0; i < 12; ++i){
69. **for**(**int** j = 0; j < 20; ++j)
70. s[change[i][j]] = tmp[change[i^1][j]];
72. a[deep] = i / 2;
73. b[deep] = i & 1?-1:1;
75. **if**(dfs(deep + 1)) **return** **true**;
76. memcpy(s, tmp, **sizeof** tmp);
77. }
78. **return** **false**;
79. }
81. **void** ida(){
82. res = geth(s);
84. **if**(!res) {printf("0\n"); **return**;}
85. res = 1;
86. **while**(res <= 5){
87. **if**(dfs(0)){
88. printf("%d\n", res);
89. **for**(**int** i = 0; i < res; ++i)
90. printf("%d %d\n", a[i], b[i]);
91. **return**;
92. }
93. res++;
94. }
95. printf("-1\n");
96. }
98. **int** main(){
99. scanf("%d", &t);
100. **while**(t--){
101. **for**(**int** i = 1; i < N; ++i)
102. s[i] = getch();
103. ida();
104. }
106. **return** 0;
107. }

七

1. **const** **int** maxn = 810;
2. **int** n, m, t;
3. **char** mp[maxn][maxn];
4. **int** vst[maxn][maxn];
5. **int** step[4][2] = {{1,0}, {-1,0}, {0,1}, {0,-1}};
7. **struct** Point{
8. **int** x, y;
9. Point(**int** xx,**int** yy):x(xx),y(yy){}
10. Point(){}
11. };
12. Point s, e, z[2];
14. **bool** check(**int** x, **int** y){
15. **if**(x < 0 || x >= n || y < 0 || y >= m || mp[x][y] == 'X')
16. **return** **false**;
17. **for**(**int** i = 0; i < 2; ++i)//利用曼哈顿距离判断是否会被鬼抓到
18. **if**(abs(x-z[i].x)+abs(y-z[i].y) <= 2\*t)
19. **return** **false**;
20. **return** **true**;
21. }
23. **void** bfs(){
24. queue<Point> qs, qe;
25. qs.push(s);
26. qe.push(e);
27. vst[s.x][s.y] = 1;
28. vst[e.x][e.y] = 2;
29. **while**(!qs.empty() || !qe.empty()){
30. t++;
31. **for**(**int** i = 0; i < 3; ++i){//3步
32. **int** cot = qs.size();
33. **while**(cot--){
34. Point p = qs.front();
35. qs.pop();
36. **if**(!check(p.x, p.y))
37. **continue**;
38. **for**(**int** j = 0; j < 4; ++j){
39. **int** cur\_x = p.x + step[j][0];
40. **int** cur\_y = p.y + step[j][1];
41. **if**(!check(cur\_x, cur\_y))
42. **continue**;
43. **if**(vst[cur\_x][cur\_y] != 1){
44. **if**(vst[cur\_x][cur\_y] == 2){
45. printf("%d\n", t);
46. **return**;
47. }
48. vst[cur\_x][cur\_y] = 1;
49. qs.push(Point(cur\_x, cur\_y));
50. }
51. }
52. }
53. }
54. **int** cot = qe.size();
55. **while**(cot--){
56. Point p = qe.front();
57. qe.pop();
58. **if**(!check(p.x, p.y))
59. **continue**;
60. **for**(**int** j = 0; j < 4; ++j){
61. **int** cur\_x = p.x + step[j][0];
62. **int** cur\_y = p.y + step[j][1];
63. **if**(!check(cur\_x, cur\_y))
64. **continue**;
65. **if**(vst[cur\_x][cur\_y] != 2){
66. **if**(vst[cur\_x][cur\_y] == 1){
67. printf("%d\n", t);
68. **return**;
69. }
70. vst[cur\_x][cur\_y] = 2;
71. qe.push(Point(cur\_x, cur\_y));
72. }
73. }
74. }
75. }
76. printf("-1\n");
77. }
79. **int** main(){
80. **int** tt;
81. scanf("%d", &tt);
82. **while**(tt--){
83. scanf("%d%d", &n, &m);
84. **int** cot = 0;
85. **for**(**int** i = 0; i < n; ++i){
86. scanf("%s", mp[i]);
87. **for**(**int** j = 0; j < m; ++j){
88. **if**(mp[i][j] == 'M')
89. s = Point(i, j);
90. **else** **if**(mp[i][j] == 'G')
91. e = Point(i, j);
92. **else** **if**(mp[i][j] == 'Z')
93. z[cot++] = Point(i, j);
94. }
95. }
96. t = 1;
97. **for**(**int** i = 0; i <= n; ++i)
98. **for**(**int** j = 0; j <= m; ++j)
99. vst[i][j] = 0;
100. **if**(!check(s.x, s.y) && !check(e.x, e.y)){
101. printf("-1\n");
102. **continue**;
103. }
104. t = 0;
105. bfs();
106. }
107. **return** 0;
108. }

八

1. **const** **int** maxn = 15;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** t, n, m;
5. **int** sx, sy, sz, ex, ey, ez;
6. **char** G[3][maxn][maxn];
7. **bool** vst[3][maxn][maxn];
8. **int** step[4][2] = {{1,0},{-1,0},{0,1},{0,-1}};
10. **struct** Point{
11. **int** x, y, z;
12. **int** d;
13. Point(){}
14. Point(**int** zz, **int** xx, **int** yy, **int** dd):z(zz),x(xx),y(yy),d(dd){}
15. };
17. **bool** check(**int** z, **int** x, **int** y){
18. **if**(x < 0 || y < 0 || x >= n || y >= m || G[z][x][y] == '\*')
19. **return** **false**;
20. **return** **true**;
21. }
23. **void** bfs(){
24. queue<Point> q;
25. q.push(Point(sz, sx, sy, 0));
26. vst[sz][sx][sy] = 1;
27. **while**(!q.empty()){
28. Point p = q.front();
29. q.pop();
30. **if**(p.z == ez && p.x == ex && p.y == ey && p.d <= t){
31. cout<<"YES"<<endl;
32. **return** ;
33. }
34. **for**(**int** i = 0; i < 4; ++i){
35. **int** cur\_x = p.x + step[i][0];
36. **int** cur\_y = p.y + step[i][1];
37. // cout<<"\*\*\*\*\*"<<endl;
38. //            cout<<setw(4)<<cur\_x<<setw(4)<<cur\_y<<endl;
39. //            cout<<setw(4)<<check(p.z, cur\_x, cur\_y)<<setw(4)<<!vst[p.z][cur\_x][cur\_y]<<setw(4)<<(p.d<t)<<endl;
40. **if**(check(p.z, cur\_x, cur\_y) && !vst[p.z][cur\_x][cur\_y] && p.d < t){
41. //                cout<<setw(4)<<p.z<<setw(4)<<p.x<<setw(4)<<p.y<<setw(4)<<p.d<<endl;
42. **if**(G[p.z][cur\_x][cur\_y] == '#'){
43. **if**(G[1-p.z][cur\_x][cur\_y] != '\*' && G[1-p.z][cur\_x][cur\_y] != '#'){
44. vst[p.z][cur\_x][cur\_y] = 1;
45. vst[1-p.z][cur\_x][cur\_y] = 1;
46. q.push(Point(1-p.z, cur\_x, cur\_y, p.d+1));
47. }
48. }
49. **else**{
50. vst[p.z][cur\_x][cur\_y] = 1;
51. q.push(Point(p.z, cur\_x, cur\_y, p.d+1));
52. }
53. }
54. }
55. }
56. cout<<"NO"<<endl;
57. }
59. **int** main(){
60. **int** T;
61. ios::sync\_with\_stdio(**false**);
62. cin >> T;
63. sx = sy = sz = 0;
64. **while**(T--){
65. cin >> n >> m >> t;
66. **for**(**int** k = 0; k < 2; ++k)
67. **for**(**int** i = 0; i < n; ++i)
68. **for**(**int** j = 0; j < m; ++j){
69. cin >> G[k][i][j];
70. vst[k][i][j] = 0;
71. **if**(G[k][i][j] == 'P'){
72. ez = k;
73. ex = i;
74. ey = j;
75. }
76. }
77. bfs();
78. }
79. **return** 0;
80. }

九

1. **const** **int** maxn = 15;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** n, m;
5. **int** Three[11];
6. **int** G[maxn][maxn];
7. **int** dp[60000][11];
8. **int** digit[60000][11];
10. **void** init(){
11. Three[0] = 1;
12. **for**(**int** i = 1; i < 11; ++i)
13. Three[i] = Three[i-1] \* 3;
14. **for**(**int** i = 0; i < Three[10]; ++i){
15. **int** tmp = i;
16. **for**(**int** j = 0; j < 10; ++j){
17. digit[i][j] = tmp%3;
18. tmp /= 3;
19. }
20. }
21. }
23. **int** main(){
24. init();
25. **while**(~scanf("%d%d", &n, &m)){
26. **for**(**int** i = 0; i <= n; ++i)
27. **for**(**int** j = 0; j <= n; ++j)
28. G[i][j] = inf;
29. **for**(**int** i = 0; i < Three[n]; ++i)
30. **for**(**int** j = 0; j < n; ++j)
31. dp[i][j] = inf;
32. **while**(m--){
33. **int** x, y, z;
34. scanf("%d%d%d", &x, &y, &z);
35. G[x-1][y-1] = G[y-1][x-1] = min(G[x-1][y-1], z);
36. }
37. **for**(**int** i = 0; i < n; ++i)
38. dp[Three[i]][i] = 0;
39. **int** ans = inf;
40. **for**(**int** i = 0; i < Three[n]; ++i){
41. **bool** flag = 1;
42. **for**(**int** j = 0; j < n; ++j){
43. **if**(digit[i][j] == 0)//只要三进制数中存在一个0，那么就说明还有点没有遍历完，就不能当做最终答案来求
44. flag = 0;
45. **if**(dp[i][j] != inf)
46. **for**(**int** k = 0; k < n; ++k)
47. **if**(G[j][k] != inf && digit[i][k] != 2)//注意这个digit[j][k]!=2,因为如果j状态在k点已经走过两次了显然是不能继续往下走的
48. dp[i+Three[k]][k] = min(dp[i+Three[k]][k], dp[i][j]+G[j][k]);
49. }
50. **if**(flag)
51. **for**(**int** j = 0; j < n; ++j)//由于是3进制，不能方便的判断一串三进制数里面是否存在0
52. ans = min(ans, dp[i][j]);
53. }
54. **if**(ans >= inf)
55. printf("-1\n");
56. **else**
57. printf("%d\n", ans);
58. }
59. **return** 0;
60. }

十

1. **const** **int** maxn=370000;
2. **int** fac[10];
3. **int** vis[370000];
4. string ans[maxn];
5. **int** Hash[10];
6. **int** step[4][3] = {{0,1,1},{0,-1,-1},{1,0,3},{-1,0,-3}};
7. **const** **int** dx[]= {0,0,1,-1};
8. **const** **int** dy[]= {1,-1,0,0};
9. **const** **int** dd[]= {1,-1,3,-3};
10. **char** go[]="lrud";
11. queue<**int**> q;
12. **void** init(){
13. fac[0]=1;
14. **for**(**int** i=1; i<10; i++)
15. fac[i]=fac[i-1]\*i;
16. }
17. **int** hashh(**int** k){
18. **int** t,ans=0;
19. **for**(**int** i=0; i<9; i++){
20. Hash[8-i]=k%10;
21. k/=10;
22. }
23. **for**(**int** i=0; i<9; i++){
24. t=0;
25. **for**(**int** j=i+1; j<9; j++)
26. **if**(Hash[i]>Hash[j])t++;
27. ans+=t\*fac[8-i];
28. }
29. **return** ans;
30. }
32. **bool** check(**int** x,**int** y){
33. **if**(x<0 || x>=3 || y<0 || y>=3)
34. **return** **false**;
35. **return** **true**;
36. }
38. **void** bfs(**int** k){
39. **while**(!q.empty())
40. q.pop();
41. **int** hp = hashh(k);
42. vis[hp] = 1;
43. q.push(k);
44. string s;
45. **while**(!q.empty()){
46. **int** p = q.front();
47. q.pop();
48. hp = hashh(p);
49. s = "";
50. **for**(**int** i = 0; i < 9; ++i){
51. s += p%10+48;
52. p /= 10;
53. }
54. reverse(s.begin(), s.end());
55. **int** pos;
56. **for**(**int** i = 0; i < 9; ++i)
57. **if**(s[i] == '9'){
58. pos = i;
59. **break**;
60. }
61. **int** x = pos / 3;
62. **int** y = pos % 3;
63. **for**(**int** i = 0; i < 4; ++i){
64. **int** cur\_x = x + step[i][0];
65. **int** cur\_y = y + step[i][1];
66. **if**(check(cur\_x, cur\_y)){
67. **int** cur\_p = 0;
68. swap(s[pos],s[pos+step[i][2]]);
69. **for**(**int** j = 0; j < 9; ++j){
70. cur\_p \*= 10;
71. cur\_p += s[j]-48;
72. }
73. **int** cur\_hp = hashh(cur\_p);
74. **if**(!vis[cur\_hp]){
75. vis[cur\_hp]=1;
76. q.push(cur\_p);
77. ans[cur\_hp] = ans[hp] + go[i];
78. }
79. swap(s[pos],s[pos+step[i][2]]);
80. }
81. }
82. }
83. }
84. **int** main(){
85. init();
86. bfs(123456789);
87. **for**(**int** i=0; i<maxn; i++)
88. reverse(ans[i].begin(),ans[i].end());
89. **char** ch[9];
90. **while**(~scanf("%s", ch)){
91. **for**(**int** i=1; i<9; i++)scanf("%s",ch+i);
92. **int** k=0;
93. **for**(**int** i=0; i<9; i++){
94. **if**(ch[i]=='x'){
95. ch[i]='9';
96. **break**;
97. }
98. }
99. **for**(**int** i=0; i<9; i++){
100. k \*= 10;
101. k += ch[i]-48;
102. }
103. **int** hk = hashh(k);
104. **if**(vis[hk])
105. cout<<ans[hk]<<endl;
106. **else**
107. cout<<"unsolvable"<<endl;
108. }
109. **return** 0;
110. }