一

1. **const** **int** maxn = 10;
2. **char** mp[maxn][maxn];
3. **int** n, k;
4. **int** cot;
5. **int** vst[maxn];
7. **int** ans;
8. **void** dfs(**int** x){
9. **if**(cot == k){//在x行之前已选够k个棋子
10. ans++;
11. **return**;
12. }
13. **if**(x >= n){//超出了棋盘
14. **return**;
15. }
16. **for**(**int** i = 0; i < n; ++i){
17. **if**(mp[x][i] == '#' && !vst[i]){
18. vst[i] = 1;
19. cot++;
20. dfs(x+1);
21. vst[i] = 0;
22. cot--;
23. }
24. }
25. dfs(x+1);
26. //当k<初始棋盘中的棋子n时,k未到最后一行提前满足
27. //令当前位置不放棋子，搜索在下一行放棋子
28. }
30. **int** main(){
31. **while**(~scanf("%d%d", &n, &k)){
32. **if**(n == -1 || k == -1)
33. **return** 0;
34. memset(vst, 0, **sizeof**(vst));
35. cot = 0;
36. ans = 0;
37. **for**(**int** i = 0; i < n; ++i){
38. scanf("%s", mp[i]);
39. }
40. dfs(0);
41. printf("%d\n", ans);
42. }
43. **return** 0;
44. }

二

1. **const** **int** maxn = 110;
2. **int** t, n, m;
3. **int** mp[maxn][maxn][maxn];
4. **int** vst[maxn][maxn][maxn];
5. **int** sz,sx,sy,ez,ex,ey;
6. **int** step[6][3] = {{0,0,1},{0,0,-1},{0,1,0},{0,-1,0},{1,0,0},{-1,0,0}};
8. **struct**  Point{
9. **int** x,y,z;
10. **int** d;
11. Point(**int** zz, **int** xx, **int** yy, **int** dis){
12. z = zz;
13. x = xx;
14. y = yy;
15. d = dis;
16. }
17. Point(){}
18. };
20. **bool** check(**int** z, **int** x, **int** y){
21. **if**(z < 0 || z >= t || x < 0 || x >= n || y < 0 || y >= m || mp[z][x][y] == '#')
22. **return** **false**;
23. **return** **true**;
24. }
26. **int** ans;
28. **void** bfs(){
29. queue<Point> q;
30. q.push(Point(sz,sx,sy,0));
31. **while**(!q.empty()){
32. Point p = q.front();
33. q.pop();
34. **if**(p.z == ez && p.x == ex && p.y == ey){
35. ans = p.d;
36. printf("Escaped in %d minute(s).\n", ans);
37. **return**;
38. }
39. **for**(**int** i = 0; i < 6; ++i){
40. **int** cur\_z = p.z + step[i][0];
41. **int** cur\_x = p.x + step[i][1];
42. **int** cur\_y = p.y + step[i][2];
43. **if**(!vst[cur\_z][cur\_x][cur\_y] && check(cur\_z, cur\_x, cur\_y)){
44. vst[cur\_z][cur\_x][cur\_y] = 1;
45. q.push(Point(cur\_z,cur\_x,cur\_y, p.d+1));
46. }
47. }
48. }
49. printf("Trapped!\n");
50. }
52. **int** main(){
53. **while**(~scanf("%d%d%d", &t, &n, &m)){
54. **if**(!t || !n || !m)
55. **return** 0;
56. getchar();
57. memset(vst, 0, **sizeof**(vst));
58. **for**(**int** k = 0; k < t; ++k){
59. **for**(**int** i = 0; i < n; ++i){
60. **for**(**int** j = 0; j < m; ++j){
61. scanf("%c", &mp[k][i][j]);
62. **if**(mp[k][i][j] == 'S'){
63. sz = k;
64. sx = i;
65. sy = j;
66. }
67. **if**(mp[k][i][j] == 'E'){
68. ez = k;
69. ex = i;
70. ey = j;
71. }
72. }
73. getchar();
74. }
75. **if**(k != t-1)
76. getchar();
77. }
78. ans = 0;
79. bfs();
80. }
81. **return** 0;
82. }

三

1. **const** **int** maxn = 1e5+10;
2. **int** n, k;
3. **int** d[maxn];
4. **int** vst[maxn];
6. **int** bfs(){
7. **if**(n == k)  //不能省略？
8. **return** 0;
9. queue<**int**> q;
10. vst[n] = 1;
11. q.push(n);
12. **while**(!q.empty()){
13. **int** p = q.front();
14. q.pop();
15. **if**(p == k){
16. **return** d[p];
17. }
18. **if**(p+1 <= maxn && !vst[p+1]){
19. q.push(p+1);
20. vst[p+1] = 1;
21. d[p+1] = d[p]+1;
22. }
23. **if**(p-1 >= 0 && !vst[p-1]){
24. q.push(p-1);
25. vst[p-1] = 1;
26. d[p-1] = d[p]+1;
27. }
28. **if**(p\*2 <= maxn && !vst[p\*2]){
29. q.push(p\*2);
30. vst[p\*2] = 1;
31. d[p\*2] = d[p]+1;
32. }
33. }
34. }
36. **int** main(){
37. scanf("%d%d", &n, &k);
38. **int** ans = bfs();
39. printf("%d", ans);
40. **return** 0;
41. }

四

1. **const** **int** maxn = 20;
2. **int** n, m;
3. **int** mp[maxn][maxn];
4. **int** flip[maxn][maxn];
5. **int** ans[maxn][maxn];
6. **int** step[5][2] = {{0,1},{0,-1},{1,0},{-1,0},{0,0}};
8. **bool** check(**int** x,**int** y){
9. **if**(x<0 || x>=n || y<0 || y>=m)
10. **return** **false**;
11. **return** **true**;
12. }
14. **bool** isFlip(**int** x, **int** y){
15. **int** color = mp[x][y];
16. **for**(**int** i = 0; i < 5; ++i){
17. **int** cur\_x = x+step[i][0];
18. **int** cur\_y = y+step[i][1];
19. **if**(flip[cur\_x][cur\_y] && check(cur\_x, cur\_y))
20. color++;
21. }
22. **return** color%2;
23. }
25. **int** getFlipNum(){
26. **int** sum = 0;
27. **for**(**int** i = 0; i < m; ++i)
28. **if**(flip[0][i])
29. sum++;
30. **for**(**int** i = 1; i < n; ++i)
31. **for**(**int** j = 0; j < m; ++j)
32. **if**(isFlip(i-1, j)){
33. flip[i][j] = 1;
34. sum++;
35. }
36. **for**(**int** i = 0; i < m; ++i)
37. **if**(isFlip(n-1, i))
38. **return** -1;
39. **return** sum;
40. }
42. **int** MIN = INT\_MAX;
44. **int** main(){
45. scanf("%d%d", &n, &m);
46. **for**(**int** i = 0; i < n; ++i)
47. **for**(**int** j = 0; j < m; ++j)
48. scanf("%d", &mp[i][j]);
49. **for**(**int** i = 0; i < 1<<m; ++i){
50. memset(flip, 0, **sizeof**(flip));
51. **for**(**int** j = 0; j < m; ++j)
52. flip[0][m-j-1] = i>>j&1;
53. **int** cot = getFlipNum();
54. **if**(cot == -1)
55. **continue**;
56. **if**(MIN > cot){
57. MIN = cot;
58. memcpy(ans, flip, **sizeof**(flip));
59. }
60. }
61. **if**(MIN == INT\_MAX)
62. printf("IMPOSSIBLE");
63. **else**{
64. **for**(**int** i = 0; i < n; ++i){
65. **if**(i)
66. printf("\n");
67. **for**(**int** j = 0; j < m; ++j){
68. **if**(j)
69. printf(" ");
70. printf("%d", ans[i][j]);
71. }
72. }
73. }
74. **return** 0;
75. }

五

1. **typedef** unsigned **long** **long** ULL;
2. **const** **int** maxn = 30;
3. **int** n;
4. **int** flag;
5. **int** a[maxn];
7. **void** dfs(ULL x, **int** len){
8. **if**(!flag)
9. **return** ;
10. **if**(x % n == 0){
11. flag = 0;
12. printf("%lld\n", x);
13. a[n] = x;
14. **return**;
15. }
16. **if**(len >= 19)
17. **return** ;
18. dfs(x\*10, len+1);
19. dfs(x\*10+1, len+1);
20. }
22. **int** main(){
23. **while**(~scanf("%d", &n)){
24. **if**(!n)
25. **return** 0;
26. flag = 1;
27. dfs(1,1);
28. }
29. **return** 0;
30. }

六

1. **const** **int** maxn = 1e5+10;
3. **int** t;
4. **int** n, m;
5. **int** d[maxn];
6. **int** vst[maxn];
8. **bool** isPrime(**int** x){
9. **for**(**int** i = 2; i < sqrt(x)+1; ++i)
10. **if**(x % i == 0)
11. **return** **false**;
12. **return** **true**;
13. }
15. **int** f(**int** a[], **int** x){
16. **int** sum = 0;
17. **for**(**int** k = 0; k < 4; ++k){
18. sum \*= 10;
19. sum += a[k];
20. }
21. **return** sum;
22. }
24. **void** bfs(){
25. queue<**int**> q;
26. q.push(n);
27. vst[n] = 1;
28. **while**(!q.empty()){
29. **int** p = q.front();
30. q.pop();
31. **if**(p == m){
32. printf("%d\n", d[p]);
33. **return**;
34. }
35. **int** a[4];
36. **int** tmp = p;
37. **for**(**int** i = 0; i < 4; ++i){
38. a[4-i-1] = tmp%10;
39. tmp /= 10;
40. }
41. **for**(**int** i = 0; i < 4; ++i){
42. **for**(**int** j = 0; j < 10; ++j){
43. **if**(!i && !j)
44. **continue**;
45. tmp = a[i];
46. a[i] = j;
47. **int** next = f(a, 4);
48. **if**(!vst[next] && isPrime(next)){
49. q.push(next);
50. vst[next] = 1;
51. d[next] = d[p] + 1;
52. }
53. a[i] = tmp;
54. }
55. }
56. }
57. }
59. **int** main(){
60. scanf("%d", &t);
61. **while**(t--){
62. memset(d, 0, **sizeof**(d));
63. memset(vst, 0, **sizeof**(vst));
64. scanf("%d%d", &n, &m);
65. bfs();
66. }
67. **return** 0;
68. }

七

1. string a, b, c;
2. set<string> s;
3. **int** n, t, m;
5. **void** dfs(){
6. string mo;
7. **int** cot = 0;
8. **while**(1){
9. mo = "";
10. **for**(**int** i = 0; i < n; ++i){
11. mo = mo + b[i] + a[i];
12. }
13. **if**(s.find(mo) == s.end())
14. s.insert(mo);
15. **else**
16. **break**;
17. cot ++;
18. **if**(mo.compare(c) == 0){
19. cout << m << " " << cot << endl;
20. **return** ;
21. }
22. a = mo.substr(0,n);
23. b = mo.substr(n,2\*n);
24. }
25. cout << m << " " << -1 << endl;
26. }
28. **int** main(){
29. cin >> t;
30. **for**(m = 1; m <= t; ++m){
31. s.clear();
32. cin >> n;
33. cin >> a >> b >> c;
34. dfs();
35. }
36. }

八

1. **int** a[5];
2. **int** c;
3. map<pair<**int**,**int**> ,**int**> mp;
4. map<pair<**int**,**int**> ,**int**> d;
5. map<**char**, string> ans;
7. **void** bfs(){
8. queue<pair<**int**,**int**> > q;
9. queue<string> path;
10. mp[make\_pair(0, 0)] = 1;
11. q.push(make\_pair(0,0));
12. path.push("");
13. **while**(!q.empty()){
14. pair<**int**,**int**> p;
15. p = q.front();
16. q.pop();
17. string s = path.front();
18. path.pop();
19. **if**(p.first == c || p.second == c){
20. cout<<d[p]<<endl;
21. **for**(**int** i = 0; i < s.size(); ++i){
22. cout<<ans[s[i]]<<endl;
23. }
24. **return** ;
25. }
26. **if**(p.first < a[0]){
27. **if**(!mp.count(make\_pair(a[0], p.second))){
28. q.push(make\_pair(a[0], p.second));
29. d[make\_pair(a[0], p.second)] = d[p] + 1;
30. mp[make\_pair(a[0], p.second)] = 1;
31. path.push(s+"1");
32. }
33. **if**(p.second){
34. **int** v1 = p.second + p.first > a[0] ? a[0] : p.second + p.first;
35. **int** v2 = p.second + p.first > a[0] ? p.second + p.first - a[0] : 0;
36. **if**(!mp.count(make\_pair(v1, v2))){
37. q.push(make\_pair(v1, v2));
38. d[make\_pair(v1, v2)] = d[p] + 1;
39. mp[make\_pair(v1, v2)] = 1;
40. path.push(s+"2");
41. }
42. }
43. }
44. **if**(p.second < a[1]){
45. **if**(!mp.count(make\_pair(p.first, a[1]))){
46. q.push(make\_pair(p.first, a[1]));
47. d[make\_pair(p.first, a[1])] = d[p] + 1;
48. mp[make\_pair(p.first, a[1])] = 1;
49. path.push(s+"3");
50. }
51. **if**(p.first){
52. **int** v2 = p.second + p.first > a[1] ? a[1] : p.second + p.first;
53. **int** v1 = p.second + p.first > a[1] ? p.second + p.first - a[1] : 0;
54. **if**(!mp.count(make\_pair(v1, v2))){
55. q.push(make\_pair(v1, v2));
56. d[make\_pair(v1, v2)] = d[p] + 1;
57. mp[make\_pair(v1, v2)] = 1;
58. path.push(s+"4");
59. }
60. }
61. }
62. **if**(p.first){
63. **if**(!mp.count(make\_pair(0, p.second))){
64. q.push(make\_pair(0, p.second));
65. d[make\_pair(0, p.second)] = d[p] + 1;
66. mp[make\_pair(0, p.second)] = 1;
67. path.push(s+"5");
68. }
69. }
70. **if**(p.second){
71. **if**(!mp.count(make\_pair(p.first, 0))){
72. q.push(make\_pair(p.first, 0));
73. d[make\_pair(p.first, 0)] = d[p] + 1;
74. mp[make\_pair(p.first, 0)] = 1;
75. path.push(s+"6");
76. }
77. }
78. }
79. cout<<"impossible"<<endl;
80. }
82. **int** main(){
83. ans['1'] = "FILL(1)";
84. ans['2'] = "POUR(2,1)";
85. ans['3'] = "FILL(2)";
86. ans['4'] = "POUR(1,2)";
87. ans['5'] = "DROP(1)";
88. ans['6'] = "DROP(2)";
89. **while**(cin>>a[0]>>a[1]>>c){
90. **if**(a[0] == c || a[1] == c){
91. cout<<1<<endl;
92. **if**(a[0] == c)
93. cout<<"FILL(1)"<<endl;
94. **else**
95. cout<<"FILL(2)"<<endl;
96. **continue**;
97. }
98. **if**(a[0] < c && a[1] < c){
99. cout<<"impossible"<<endl;
100. **continue**;
101. }
102. d.clear();
103. mp.clear();
104. bfs();
105. }
106. **return** 0;
107. }

九

1. **const** **int** INF = 0x3f3f3f3f;
2. **const** **int** N = 20;
3. **const** **int** dir[4][2] = { {0, 1}, {0, -1}, {1, 0}, {-1, 0}};
5. **struct** Point {
6. **int** x, y;
7. Point() {}
8. Point(**int** a, **int** b) { x = a; y = b; }
9. };
10. **int** m, n, d[N][N], tmp;
11. **char** g[N][N];
13. **void** init() {
14. memset(g, 0, **sizeof**(g));
15. tmp = 0;
17. scanf("%d%d%\*c", &m, &n);
19. **for** (**int** i = 0; i < m; i++) {
20. gets(g[i]);
21. **for** (**int** j = 0; j < n; j++)
22. **if** (g[i][j] == '#')
23. tmp++;
24. }
25. }
27. **int** bfs(**int** x1, **int** y1, **int** x2, **int** y2) {
28. memset(d, INF, **sizeof**(d));
30. d[x1][y1] = d[x2][y2] = 0;
32. Point k, c;
34. queue<Point> q;
35. q.push(Point(x1, y1));
36. q.push(Point(x2, y2));
38. **while** ( !q.empty() ) {
39. k = q.front(); q.pop();
41. **for** (**int** i = 0; i < 4; i++) {
42. c.x = k.x + dir[i][0]; c.y = k.y + dir[i][1];
43. **if** (c.x < 0 || c.x >= m || c.y < 0 || c.y >= n) **continue**;
44. **if** (g[c.x][c.y] != '#') **continue**;
46. **if** (d[c.x][c.y] > d[k.x][k.y] + 1) {
47. d[c.x][c.y] = d[k.x][k.y] + 1;
48. q.push(c);
49. }
50. }
51. }
53. **int** ans = 0;
54. **for** (**int** i = 0; i < m; i++) {
55. **for** (**int** j = 0; j < n; j++) **if** (g[i][j] == '#') {
56. ans = max(ans, d[i][j]);
57. }
58. }
59. **return** ans;
60. }

63. **int** solve() {
64. **if** (tmp <= 2) **return** 0;
65. **int** ans = INF;
66. **for** (**int** i = 0; i < m; i++) {
67. **for** (**int** j = 0; j < n; j++) **if** (g[i][j] == '#') {
68. **for** (**int** k = 0; k < m; k++) {
69. **for** (**int** t = 0; t < n; t++) {
70. **if** (k == i && t <= j) **continue**;
71. **if** (g[k][t] == '#')
72. ans = min(ans, bfs(i, j, k, t));
73. }
74. }
75. }
76. }
77. **return** ans == INF ? -1 : ans;
78. }
80. **int** main () {
81. **int** cas;
82. scanf("%d", &cas);
83. **for** (**int** i = 1; i <= cas; i++) {
84. init();
85. printf("Case %d: %d\n", i, solve());
86. }
87. **return** 0;
88. }

十

1. **const** **int** maxn = 1e3+10;
2. **const** **int** inf = 0x3f3f3f3f;
3. **int** n, m;
4. **int** sx, sy;
5. **int** fx, fy;
6. **char** G[maxn][maxn];
7. **int** vst[maxn][maxn];
8. **int** Time[maxn][maxn];
9. **int** curT[maxn][maxn];
10. **int** step[4][2] = {{1,0},{-1,0},{0,1},{0,-1}};
12. **struct** Point{
13. **int** x, y;
14. Point(**int** xx, **int** yy):x(xx),y(yy){}
15. Point(){}
16. };
18. **bool** check(**int** x, **int** y){
19. **if**(x<0 || x>=n || y<0 || y>=m || vst[x][y])
20. **return** **false**;
21. **return** G[x][y] == '.';
22. }
24. **bool** onBorder(**int** x, **int** y){
25. **if**(x == 0 || x == n-1 || y == 0 || y == m-1)
26. **return** **true**;
27. **return** **false**;
28. }
30. **void** bfs\_fire(queue<Point> fireQ){
31. queue<Point> q;
32. **while**(!fireQ.empty()){
33. q.push(fireQ.front());
34. fireQ.pop();
35. }
36. **while**(!q.empty()){
37. Point p = q.front();
38. q.pop();
39. **for**(**int** i = 0; i < 4; ++i){
40. **int** cur\_x = p.x + step[i][0];
41. **int** cur\_y = p.y + step[i][1];
42. **if**(check(cur\_x, cur\_y)){
43. vst[cur\_x][cur\_y] = 1;
44. Time[cur\_x][cur\_y] = Time[p.x][p.y] + 1;
45. q.push(Point(cur\_x, cur\_y));
46. }
47. }
48. }
49. }
51. **void** bfs(){
52. queue<Point> q;
53. q.push(Point(sx, sy));
54. curT[sx][sy] = 0;
55. vst[sx][sy] = 1;
56. **while**(!q.empty()){
57. Point p = q.front();
58. q.pop();
59. **if**(onBorder(p.x, p.y)){
60. printf("%d\n", curT[p.x][p.y]+1);
61. **return** ;
62. }
63. **for**(**int** i = 0; i < 4; ++i){
64. **int** cur\_x = p.x + step[i][0];
65. **int** cur\_y = p.y + step[i][1];
66. **if**(check(cur\_x, cur\_y)){
67. **if**(curT[p.x][p.y] + 1 < Time[cur\_x][cur\_y]){
68. curT[cur\_x][cur\_y] = curT[p.x][p.y] + 1;
69. vst[cur\_x][cur\_y] = 1;
70. q.push(Point(cur\_x, cur\_y));
71. }
72. }
73. }
74. }
75. printf("IMPOSSIBLE\n");
76. }
78. **int** main(){
79. **int** t;
80. ios::sync\_with\_stdio(**false**);
81. scanf("%d", &t);
82. **while**(t--){
83. scanf("%d%d", &n, &m);
84. **for**(**int** i = 0; i < n; ++i)
85. **for**(**int** j = 0; j < m; ++j){
86. vst[i][j] = 0;
87. Time[i][j] = inf;
88. curT[i][j] = 0;
89. }
90. queue<Point> q;
91. **for**(**int** i = 0; i < n; ++i){
92. scanf("%s", G[i]);
93. **for**(**int** j = 0; j < m; ++j){
94. **if**(G[i][j] == 'J'){
95. sx = i;
96. sy = j;
97. }
98. **if**(G[i][j] == 'F'){
99. vst[i][j] = 1;
100. Time[i][j] = 0;
101. q.push(Point(i, j));
102. }
103. }
104. }
105. bfs\_fire(q);
106. **for**(**int** i = 0; i < n; ++i)
107. **for**(**int** j = 0; j < m; ++j)
108. vst[i][j] = 0;
109. bfs();
110. }
111. **return** 0;
112. }

十一

1. **const** **int** n = 5;
2. **int** step[][2]={{0,-1},{-1,0},{0,1},{1,0}};
3. **int** vst[100][100];
4. **int** mp[100][100];
6. **bool** check(**int** x,**int** y){
7. **if**(x < 0 || x >= n || y < 0 || y >= n)
8. **return** **false**;
9. **return** **true**;
10. }
12. **struct** Point{
13. **int** x, y;
14. Point(**int** xx, **int** yy):x(xx), y(yy){}
15. Point(){}
16. };

19. **void** bfs(){
20. queue<Point> q;
21. q.push(Point(0, 0));
22. queue<string> qq;
23. qq.push("00");
24. **while**(!q.empty()){
25. Point p = q.front();
26. q.pop();
27. string pp = qq.front();
28. qq.pop();
29. **if**(p.x == n-1 && p.y == n-1){
30. **for**(**int** i = 0; i < pp.size(); i += 2){
31. **if**(i)
32. cout<<endl;
33. cout<<"("<<pp[i]<<", "<<pp[i+1]<<")";
34. }
35. **return** ;
36. }
37. **for**(**int** i = 0; i < 4; i++){
38. **int** cur\_x = p.x + step[i][0];
39. **int** cur\_y = p.y + step[i][1];
40. **if**(check(cur\_x,cur\_y) && !vst[cur\_x][cur\_y] && !mp[cur\_x][cur\_y] ){
41. vst[cur\_x][cur\_y] = 1;
42. q.push(Point(cur\_x, cur\_y));
43. string mo = pp + **char**(cur\_x + '0') + **char**(cur\_y + '0');
44. qq.push(mo);
45. }
46. }
47. }
48. }
50. **int** main(){
51. **for**(**int** i = 0; i < n; i++)
52. **for**(**int** j =0; j < n; j++){
53. cin >> mp[i][j];
54. vst[i][j] = 0;
55. }
56. vst[0][0] = 1;
57. bfs();
58. }

十二

1. **int** n, m;
2. **char** mp[maxn][maxn];
3. **bool** vst[maxn][maxn];
4. **int** ans = 0;
5. **int** step[9][2] = {{0,1},{0,-1},{1,0},{-1,0},{1,1},{1,-1},{-1,1},{-1,-1}};
7. **bool** check(**int** x, **int** y){
8. **if**(x < 0 || x >= n || y < 0 || y >= m)
9. **return** **false**;
10. **if**(vst[x][y])
11. **return** **false**;
12. **return** **true**;
13. }
15. **void** dfs(**int** x, **int** y){
16. **for**(**int** i = 0; i < 9; ++i){
17. **int** cur\_x = x + step[i][0];
18. **int** cur\_y = y + step[i][1];
19. **if**(check(cur\_x, cur\_y) && mp[cur\_x][cur\_y] == '@'){
20. vst[cur\_x][cur\_y] = 1;
21. dfs(cur\_x, cur\_y);
22. }
23. }
24. }
26. **int** main(){
27. **while**(~scanf("%d%d", &n, &m)){
28. **if**(!n || !m)
29. **return** 0;
30. ans = 0;
31. **for**(**int** i = 0; i < n; i++)
32. **for**(**int** j = 0; j < m; j++){
33. cin >> mp[i][j];
34. vst[i][j] = 0;
35. }
36. **for**(**int** i = 0; i < n; ++i)
37. **for**(**int** j = 0; j < m; ++j){
38. **if**(mp[i][j] == '@' && !vst[i][j]){
39. dfs(i, j);
40. ans++;
41. }
42. }
43. cout << ans << endl;
44. }
45. **return** 0;
46. }

十三

1. **const** **int** maxn = 110;
2. **int** s, n, m;
3. **int** a[10];
4. **int** vst[maxn][maxn];
6. **struct** Point{
7. **int** a[3], d;
8. Point(**int** x, **int** y, **int** z, **int** dd){
9. a[0] = x;
10. a[1] = y;
11. a[2] = z;
12. d = dd;
13. }
14. Point(){}
15. };
17. **void** bfs(){
18. queue<Point> q;
19. q.push(Point(s, 0, 0, 0));
20. vst[s][0] = 1;
21. **while**(!q.empty()){
22. Point p = q.front();
23. q.pop();
24. **if**( (p.a[0] == p.a[1] && !p.a[2]) || (p.a[1] == p.a[2] && !p.a[0]) || (p.a[0] == p.a[2] && !p.a[1]) ){
25. printf("%d\n", p.d);
26. **return** ;
27. }
28. **for**(**int** i = 0; i < 3; ++i)
29. **for**(**int** j = 0; j < 3; ++j){
30. **if**(i != j){
31. **int** b[3];
32. b[j] = p.a[i]+p.a[j]>=a[j] ? a[j] : p.a[i]+p.a[j];
33. b[i] = p.a[i]+p.a[j]>=a[j] ? p.a[i]-(a[j]-p.a[j]) : 0;
34. **for**(**int** k = 0; k < 3; ++k)
35. **if**(i != k && j != k)
36. b[k] = p.a[k];
37. **if**(!vst[b[0]][b[1]]){
38. vst[b[0]][b[1]] = 1;
39. q.push(Point(b[0], b[1], b[2], p.d+1));
40. }
41. }
42. }
44. }
45. printf("NO\n");
46. }
48. **int** main(){
49. **while**(~scanf("%d%d%d", &s, &n, &m)){
50. **if**(!s || !n || !m)
51. **return** 0;
52. **for**(**int** i = 0; i <= s; ++i)
53. **for**(**int** j = 0; j <= s; ++j)
54. vst[i][j] = 0;
55. a[0] = s; a[1] = n; a[2] = m;
56. bfs();
57. }
58. **return** 0;
59. }

十四

1. **const** **int** maxn = 210;
2. **const** **int** inf = 0x3f3f3f3f - 100;
3. **struct** Point{
4. **int** x, y, d;
5. Point(**int** xx, **int** yy, **int** dd):x(xx),y(yy),d(dd){}
6. Point(){}
7. };
8. **int** n, m;
9. **char** mp[maxn][maxn];
10. **bool** vst[maxn][maxn];
11. **int** sx, sy;
12. **int** ex, ey;
13. **int** step[4][2] = {{0,1},{0,-1},{1,0},{-1,0}};
14. vector<Point> KCF;
15. **int** ds[maxn][maxn];
16. **int** de[maxn][maxn];
18. **bool** check(**int** x, **int** y){
19. **if**(x<0 || x>=n || y<0 || y>=m || vst[x][y] || mp[x][y] == '#')
20. **return** **false**;
21. **return** **true**;
22. }
24. **void** bfsS(){
25. queue<Point> q;
26. q.push(Point(sx, sy, 0));
27. vst[sx][sy] = 1;
28. **while**(!q.empty()){
29. Point p = q.front();
30. q.pop();
31. **if**(mp[p.x][p.y] == '@'){
32. ds[p.x][p.y] = p.d;
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<<p.x<<"    "<<p.y<<"    check:"<<check(cur\_x, cur\_y)<<"    mp:"<< (mp[cur\_x][cur\_y] != '#')<<endl;
38. **if**(check(cur\_x, cur\_y)){
39. q.push(Point(cur\_x, cur\_y, p.d+1));
40. vst[cur\_x][cur\_y] = 1;
41. }
42. }
43. }
44. }
46. **void** bfsE(){
47. queue<Point> q;
48. q.push(Point(ex, ey, 0));
49. vst[ex][ey] = 1;
50. **while**(!q.empty()){
51. Point p = q.front();
52. q.pop();
53. **if**(mp[p.x][p.y] == '@'){
54. de[p.x][p.y] = p.d;
55. }
56. **for**(**int** i = 0; i < 4; ++i){
57. **int** cur\_x = p.x + step[i][0];
58. **int** cur\_y = p.y + step[i][1];
59. // cout<<p.x<<"    "<<p.y<<"    check:"<<check(cur\_x, cur\_y)<<"    mp:"<< (mp[cur\_x][cur\_y] != '#')<<endl;
60. **if**(check(cur\_x, cur\_y)){
61. q.push(Point(cur\_x, cur\_y, p.d+1));
62. vst[cur\_x][cur\_y] = 1;
63. }
64. }
65. }
66. }
68. **int** main(){
69. **while**(cin>>n>>m){
70. KCF.clear();
71. **for**(**int** i = 0; i < n; ++i)
72. **for**(**int** j = 0; j < m; ++j){
73. cin>>mp[i][j];
74. vst[i][j] = 0;
75. ds[i][j] = inf;
76. de[i][j] = inf;
77. **if**(mp[i][j] == 'Y'){
78. sx = i; sy = j;
79. }
80. **if**(mp[i][j] == 'M'){
81. ex = i; ey = j;
82. }
83. **if**(mp[i][j] == '@'){
84. KCF.push\_back(Point(i, j, 0));
85. }
86. }
87. bfsS();
88. **for**(**int** i = 0; i < n; ++i)
89. **for**(**int** j = 0; j < m; ++j)
90. vst[i][j] = 0;
91. bfsE();
92. **int** MIN = INT\_MAX;
93. **for**(**int** i = 0; i < KCF.size(); ++i)
94. MIN = min(MIN, ds[KCF[i].x][KCF[i].y] + de[KCF[i].x][KCF[i].y]);
95. cout<<MIN\*11<<endl;
96. }
97. **return** 0;
98. }