

A 矩形数量

```
1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  #include<algorithm>
5  #define N 50
6  using namespace std;
7  struct use{
8      int x,y;
9      bool operator < (const use &ano)const{
10         return (x==ano.x ? y<ano.y : x<ano.x);
11     }
12     bool operator == (const use &ano)const{
13         return x==ano.x && y==ano.y;
14     }
15 }point[N];
16 int main(){
17     // freopen("test.in","r",stdin);
18     //freopen("test.out","w", stdout);
19
20     int t, i, j, p, q, n, tot;
21     scanf("%d",&t);
22     while(t--){
23         scanf("%d", &n);
24         for (i=0;i<n;++i)
25             scanf("%d%d",&point[i].x, &point[i].y);
26         sort(point, point+n);
27         n=unique(point,point+n)-point;
28         //cout<<n<<endl;
29
30         tot = 0;
31         for (i=0;i<n;++i)
32             for (j=i+1;j<n;++j){
33                 if (point[i].x != point[j].x)
34                     break;
35                 if (point[i].y == point[j].y)
36                     continue;
37                 for (p=j+1;p<n;++p){
38                     if (point[p].y != point[i].y || point[p].x == point[i].x)
39                         continue;
40                     for (q=p+1;q<n;++q){
41                         if (point[p].x != point[q].x)
42                             break;
43                         if (point[p].y == point[q].y || point[q].y != point[j].y)
44                             continue;
45                         tot += 1;
46                     }
47                 }
48             }
49
50         printf("%d\n", tot);
51     }
52 }
```

B 邮票收集

```
1  #include <iostream>
```

```

2  #include <stdio>
3  #include <cstring>
4  #include <algorithm>
5  using namespace std;
6  int n, m, a[105], f[1005];
7  int main(){
8      while (scanf("%d%d", &n, &m) && (n || m)){
9          for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
10         f[0] = 0;
11         for (int i = 1; i <= m; i++) f[i] = 1e9;
12         for (int i = 1; i <= n; i++)
13             for (int j = a[i]; j <= m; j++)
14                 f[j] = min(f[j], f[j - a[i]] + 1);
15         printf("%d\n", (f[m] == 1e9) ? -1 : f[m]);
16     }
17     return 0;
18 }

```

C 田忌赛马

```

1  #include <stdio.h>
2  #include <set>
3  #include <iostream>
4  using namespace std;
5  int N, X;
6  int enemy[12];
7  int ours[12];
8  void test(int p, int win){
9      //cout<<"--"<<p<<" "<<win<<" "<<ours[0]<<" "<<ours[1]<<endl;
10     int cur_win=win;
11     for(int i=1;i<=N;i++){
12         int used=0;
13         for(int j=0;j<p;j++){
14             if(ours[j]==i){
15                 used=1;
16                 break;
17             }
18         }
19         if(used!=0)
20             continue;
21         ours[p]=i;
22         if(i>=enemy[p]+X)
23             cur_win=win+1;
24         if(p==N-1){
25             if(cur_win*2>N){
26                 for(int j=0;j<N;j++){
27                     printf("%d ", ours[j]);
28                 }
29                 printf("\n");
30             }
31             return;
32         }
33         test(p+1, cur_win);
34     }
35 }
36
37 }
38 int main(){

```

```

39     cin>>N>>X;
40     for(int i=0;i<N;++i){
41         cin>>enemy[i];
42     }
43     test(0, 0);
44 }

```

D 漫漫回国路

```

1  #include<iostream>
2  #include<cstdio>
3
4  using namespace std;
5
6  const int MAX = 101;
7  int map[MAX][MAX];
8  bool lib[MAX]; //标记该点是否已经来过
9  int n, x0, y0;
10
11 bool DFS(int current)
12 {
13     if (current == n-1) return true;
14     bool flag = false;
15
16     for(int ny = 0; ny < n; ny++){
17         if (!lib[ny] && map[current][ny] > 0)
18         {
19             lib[ny] = 1; //走过了就不能再来
20             flag = DFS(ny);
21         }
22
23         if (flag) //找到终点, 返回真
24             return true;
25     }
26     return false;
27 }
28
29
30 int main()
31 {
32     int k;
33     cin >> k;
34     while (k-- > 0)
35     {
36         cin >> n;
37         for (int i=0; i<n; i++)
38         {
39             lib[i] = 0;
40             for (int j=0; j<n; j++)
41             {
42                 cin >> map[i][j];
43             }
44         }
45
46         if (DFS(0))
47             cout << "YES" << endl;
48         else
49             cout << "NO" << endl;

```

```

50     }
51
52     return 0;
53 }
54

```

E Project Summer

```

1  #include <stdio>
2  #include <cstring>
3  #include <algorithm>
4  #include <iostream>
5  #include <vector>
6  #include <cassert>
7  #define printf(...) fprintf(stderr, __VA_ARGS__)
8  using namespace std;
9
10 int T, n, m, sx, sy, tx, ty;
11 char maps[105][105];
12 vector<pair<int,int> > gates[35];
13 int qx[10005],qy[10005],r,dis[105][105];
14
15 pair<int,int> other(int x,int y) {
16     char p=maps[x][y];
17     if('a' <= p && p <='z') {
18         if(gates[p-'a'][0]==make_pair(x,y))
19             return gates[p-'a'][1];
20         else
21             return gates[p-'a'][0];
22     } else {
23         return make_pair(-1, -1);
24     }
25 }
26
27 int valid(int x,int y) {
28     if(x < 0 || x >= n || y < 0 || y >= m)
29         return 0;
30     if(maps[x][y]!='#')
31         return 0;
32     return 1;
33 }
34
35 void push(int x,int y,int d) {
36     if(valid(x,y) && dis[x][y]==-1) {
37         ++r; qx[r]=x; qy[r]=y; dis[x][y]=d;
38     }
39 }
40
41 void solve() {
42     scanf("%d%d", &n, &m);
43     assert(1 <= n && n <= 100);
44     assert(1 <= m && m <= 100);
45     for (int i = 0; i < n; ++i)
46         scanf("%s", maps[i]);
47     sx=sy=tx=ty=-1;
48     for(int w=0;w<26;w++)
49         gates[w].clear();
50     for(int i=0;i<n;i++) for(int j=0;j<m;j++) dis[i][j]=-1;

```

```

51     for(int i=0;i<n;i++)
52         for(int j=0;j<m;j++) {
53             char cur = maps[i][j];
54             assert(cur == 'B' || cur == 'I' || ('a' <= cur && cur <= 'z') || cur == '.' || cur == '#');
55             if(cur=='B')
56                 sx = i, sy = j;
57             if(cur=='I')
58                 tx = i, ty = j;
59             if('a' <= cur && cur <= 'z')
60                 gates[cur - 'a'].push_back(make_pair(i, j));
61         }
62
63     //printf("%d %d %d %d\n",sx,sy,tx,ty);
64     assert(valid(sx,sy)==1);
65     assert(valid(tx,ty)==1);
66     for(int w=0;w<26;w++)
67         assert(gates[w].size()==0||gates[w].size()==2);
68
69     qx[r]=sx; qy[l]=sy; dis[sx][sy]=0;
70
71     static int tes=0;
72     for(int l=1;l<=r;l++) {
73         int ux=qx[l],uy=qy[l],d=dis[ux][uy]+1;
74         if(ux==tx && uy==ty) {
75             printf("Case #d: %d\n",++tes,d-1);
76             return;
77         }
78         pair<int,int> vv = other(ux, uy);
79         push(ux-1,uy,d);
80         push(ux+1,uy,d);
81         push(ux,uy-1,d);
82         push(ux,uy+1,d);
83         push(vv.first, vv.second, d);
84     }
85     printf("Case #d: %d\n",++tes,-1);
86     return;
87 }
88
89 int main(int argc, char const *argv[])
90 {
91
92     scanf("%d", &T);
93     for (int i = 0; i < T; ++i)
94         solve();
95     return 0;
96 }

```

F 物资打包

```

1  #include <iostream>
2  #include <cmath>
3  #include <cstring>
4  #include <cstdio>
5  #include <cassert>
6  #include <algorithm>
7  using namespace std;
8  int n,m;
9  const int N=11000;

```

```

10  int A[N];
11  long long check(int now){
12      long long ans=0;
13      for (int i=1;i<=n;i++){
14          ans+=(A[i]-1)/now+1;
15      }
16      return ans;
17  }
18  int main(){
19      scanf("%d%d",&n,&m); assert(n>=100&&n<=1000&&m>=n&&m<=10000);
20      for (int i=1;i<=n;i++){
21          scanf("%d",&A[i]); assert(1000000<=A[i]&&A[i]<=10000000);
22      }
23      int l=1,r=0,ans=0;
24      for (int i=1;i<=n;i++) r=max(r,A[i]);
25      r+=1;
26      while (l<r){
27          int mid=l+r>>1;
28          if (check(mid)<=m){
29              ans=mid; r=mid;
30          } else l=mid+1;
31      }
32      printf("%d\n",ans);
33      return 0;
34  }

```

G 删除数字

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  using namespace std;
6
7
8  int f[1005][1005], n, a[1005];
9  int main(){
10     int tc;
11     scanf("%d", &tc);
12     while (tc --){
13         scanf("%d", &n);
14         for (int i = 1;i <= n;i ++){
15             scanf("%d", &a[i]);
16             for (int i = 1;i <= n;i ++){
17                 for (int j = 1;j <= i;j ++){
18                     if (a[i] == j) f[i][j] = max(f[i - 1][j], f[i - 1][j - 1] + 1);
19                     else f[i][j] = max(f[i - 1][j], f[i - 1][j - 1]);
20                 }
21             }
22             int ans = 0;
23             for (int i = 1;i <= n;i ++){
24                 ans = max(ans, f[n][i]);
25             }
26             printf("%d\n", ans);
27         }
28     }
29     return 0;
30 }

```

H 最长的环

```
1  /*
2     大概思路, DFS 找出所有 0 的连通块, 然后判断连通块边上的 1 是否满足构成一个环
3     一些细节可能实现时候要注意一下。
4     时间复杂度  $O(N*M)$ 。
5  */
6  #include <vector>
7  #include <list>
8  #include <map>
9  #include <set>
10 #include <queue>
11 #include <deque>
12 #include <stack>
13 #include <bitset>
14 #include <algorithm>
15 #include <functional>
16 #include <numeric>
17 #include <utility>
18 #include <sstream>
19 #include <iostream>
20 #include <iomanip>
21 #include <cstdio>
22 #include <cmath>
23 #include <cstdlib>
24 #include <ctime>
25 #include <cstring>
26 #include <cassert>
27
28 using namespace std;
29
30 typedef long long LL;
31 typedef pair<int, int> PII;
32 #define MP make_pair
33 #define FOR(v,p,k) for(int v=p;v<=k;++v)
34 #define FORD(v,p,k) for(int v=p;v>=k;--v)
35 #define REP(i,n) for(int i=0;i<(n);++i)
36 #define VAR(v,i) __typeof(i) v=(i)
37 #define FORE(i,c) for(__typeof(c.begin()) i=(c.begin());i!=(c).end();++i)
38 #define PB push_back
39 #define ST first
40 #define ND second
41 #define SIZE(x) (int)x.size()
42 #define ALL(c) c.begin(),c.end()
43 #define ZERO(x) memset(x,0,sizeof(x))
44
45 const int N = 1005;
46
47 int rows, cols;
48
49 char t[N][N];
50
51 bool considered[N][N];
52 bool visited[N][N];
53
54 int dy[] = {1, 0, -1, 0, 1, 1, -1, -1};
55 int dx[] = {0, 1, 0, -1, -1, 1, -1, 1};
56
57 bool isCycle(vector<PII>& v) {
```

```

58     FORE (it, v) {
59         visited[it->ST][it->ND] = false;
60         considered[it->ST][it->ND] = true;
61     }
62     int y = v[0].ST;
63     int x = v[0].ND;
64     int nVisited = 0;
65     while (y != -1) {
66         int ne = 0;
67         int nxy = -1, nxx = -1;
68         for (int d = 0; d < 4; ++d) {
69             int cy = y + dy[d];
70             int cx = x + dx[d];
71             if (considered[cy][cx]) {
72                 ++ne;
73                 if (!visited[cy][cx]) {
74                     nxy = cy;
75                     nxx = cx;
76                 }
77             }
78         }
79         if (ne != 2) {
80             break;
81         }
82         visited[y][x] = true;
83         ++nVisited;
84         y = nxy;
85         x = nxx;
86     }
87     FORE (it, v) {
88         visited[it->ST][it->ND] = false;
89         considered[it->ST][it->ND] = false;
90     }
91     return nVisited == (int) v.size();
92 }
93
94 vector<PII> boundary;
95 bool bad;
96
97 void dfs(int y, int x) {
98     visited[y][x] = true;
99     for (int d = 0; d < 8; ++d) {
100         int cy = y + dy[d];
101         int cx = x + dx[d];
102         if (t[cy][cx] == '0' && !visited[cy][cx]) {
103             dfs(cy, cx);
104         } else if (t[cy][cx] == '1' && !visited[cy][cx]) {
105             visited[cy][cx] = '1';
106             boundary.PB(MP(cy, cx));
107         } else if (t[cy][cx] != '0' && t[cy][cx] != '1') {
108             bad = true;
109         }
110     }
111 }
112
113 int main() {
114     //freopen("data.in", "r", stdin);
115     //freopen("data.out", "w", stdout);

```



```

116     int T;
117     cin >> T;
118     while (T-->0) {
119         ZERO(t);
120         ZERO(considered);
121         ZERO(visited);
122         scanf("%d %d", &rows, &cols);
123         for (int i = 1; i <= rows; ++i) {
124             scanf("%s", t[i] + 1);
125         }
126         int res = 0;
127         for (int i = 1; i <= rows; ++i) {
128             for (int j = 1; j <= cols; ++j) {
129                 if (i < rows && j < cols && t[i][j] == '1' && t[i + 1][j] == '1' && t[i][j + 1] == '1' && t[i + 1][j + 1] == '1') {
130                     res = max(res, 4);
131                 }
132                 if (t[i][j] == '0' && !visited[i][j]) {
133                     bad = false;
134                     boundary.clear();
135                     dfs(i, j);
136                     if (!bad && isCycle(boundary)) {
137                         res = max(res, (int) boundary.size());
138                     }
139                     FORE (it, boundary) {
140                         visited[it->ST][it->ND] = false;
141                     }
142                 }
143             }
144         }
145         printf("%d\n", res);
146     }
147 }

```