

# 2025暑期集训第7场

## T1.填数

- 从 $(1, 2)$ 到 $(n, n)$ 逐步填写新的数字，新填写的数字与其左方和上方的数的和不能为质数。

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 15;
5  int a[N][N];
6  bool vis[N * N];
7  int n;
8  bool isp[255];
9
10 bool prime(int n)
11 {
12     if(n == 1) return false;
13     for(int i = 2; i * i <= n; ++i) if (n % i == 0) return false;
14     return true;
15 }
16
17 bool dfs(int x, int y)
18 {
19     if(x == n + 1) return true;
20     for(int i = 1; i <= n * n; ++i)
21     {
22         if(vis[i]) continue;
23         if(x != 1 && !isp[i + a[x - 1][y]]) continue;
24         if(y != 1 && !isp[i + a[x][y - 1]]) continue;
25         a[x][y] = i, vis[i] = 1;
26         if(y < n) if(dfs(x, y + 1)) return true;
```

```

27         if(y == n) if(dfs(x + 1, 1)) return true;
28         a[x][y] = 0, vis[i] = 0;
29     }
30     return false;
31 }
32
33 int main()
34 {
35     scanf("%d", &n);
36     for(int i = 1; i <= 2 * n * n - 1; ++i)
37         isp[i] = prime(i);
38     a[1][1] = 1, vis[1] = 1;
39     bool res = dfs(1, 2);
40     if(!res) puts("NO");
41     else
42     {
43         for(int i = 1; i <= n; ++i)
44         {
45             for(int j = 1; j <= n; ++j)
46                 printf("%d ", a[i][j]);
47             puts("");
48         }
49     }
50     return 0;
51 }

```

- 优化：要求相邻之和为质数，那么直接枚举质数，判断能不能拆出新填的数字。

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 15;
5  int a[N][N];
6  bool vis[N * N];
7  int n;
8  bool isp[255];
9  int p[255], m;
10
11 bool prime(int n)
12 {
13     if(n == 1) return false;
14     for(int i = 2; i * i <= n; ++i) if (n % i == 0) return false;
15     return true;
16 }
17
18 bool dfs(int x, int y)
19 {
20     if(x == n + 1) return true;
21     int t = max(a[x - 1][y], a[x][y - 1]);
22     // 枚举两数之和(质数)
23     for(int j = 1; j <= m; ++j)
24     {
25         if(p[j] <= t) continue;
26         int i = p[j] - t;
27         if(i > n * n) break;
28         if(vis[i]) continue;
29         if(x != 1 && !isp[i + a[x - 1][y]]) continue;
30         if(y != 1 && !isp[i + a[x][y - 1]]) continue;
31         a[x][y] = i, vis[i] = 1;
32         if(y < n) if(dfs(x, y + 1)) return true;
33         if(y == n) if(dfs(x + 1, 1)) return true;
34         a[x][y] = 0, vis[i] = 0;
35     }
36 }
```

```
36     return false;
37 }
38
39 int main()
40 {
41     scanf("%d", &n);
42     for(int i = 1; i <= 2 * n * n - 1; ++i)
43         if(isp[i] = prime(i)) p[++m] = i;
44     a[1][1] = 1, vis[1] = 1;
45     bool res = dfs(1, 2);
46     if(!res) puts("NO");
47     else
48     {
49         for(int i = 1; i <= n; ++i)
50         {
51             for(int j = 1; j <= n; ++j)
52                 printf("%d ", a[i][j]);
53             puts("");
54         }
55     }
56     return 0;
57 }
```

## T2.倒水

- BFS
- 队列中记录三个水壶的水量、倒水步数即可。
- 可以定义标记数组 `vis[110][110]` 来标记  $A, B$  壶中水的状态，因为三个壶的水总量是固定的。

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int C = 3;
5  const int N = 110;
6  struct Node
7  {
8      int v[C], step;
9  };
10 int c[C], d;
11 bool vis[N][N];
12
13 int bfs()
14 {
15     queue<Node> q; q.push({{c[0], 0, 0}, 0});
16     vis[c[0]][c[1]] = true;
17     while(!q.empty())
18     {
19         Node x = q.front(); q.pop();
20         if(x.v[0] == d || x.v[1] == d || x.v[2] == d) return x.step;
21         for(int i = 0; i < C; ++i)
22             for(int j = 0; j < C; ++j)
23             {
24                 if(i == j) continue;
25                 // i杯没有水 j杯水满
26                 if(x.v[i] == 0 || x.v[j] == c[j]) continue;
27                 Node y = {{x.v[0], x.v[1], x.v[2]}, x.step + 1};
28                 int t = min(x.v[i], c[j] - x.v[j]);
```

```
29         y.v[i] -= t, y.v[j] += t;
30         if(vis[y.v[i]][y.v[j]]) continue; // 水的状态出现过
31         q.push(y), vis[y.v[i]][y.v[j]] = 1;
32     }
33 }
34 return -1;
35 }
36
37 int main()
38 {
39     for(int i = 0; i < C; ++i) scanf("%d", &c[i]);
40     scanf("%d", &d);
41     int ans = bfs();
42     if(ans != -1) printf("%d\n", ans);
43     else puts("false");
44     return 0;
45 }
```

## T3.Power Calculus

- 从 $x^1$ 开始，每次可以通过乘以/除以历史上生成的指数，可以生成新的指数，直到生成 $x^n$ 为止。
- 因为要找到最短的操作序列，所以理论上需要遍历所有的方案，那么生成树的深度可能非常大。
- 故而可以利用迭代加深，从0开始枚举允许的操作序列长度，判断能否得出答案。
- 同时 $n \leq 1000$ ，可以用数组进行记忆化。

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int ans[1010];
5  int n, deep;
6  int a[10010];
7
8  bool dfs(int x, int step)
9  {
10     // (a^x)不断自己乘自己 < a^n
11     if((x << (deep - step)) < n) return false;
12     if(step > deep) return false;
13     if(x == n) return true;
14     a[step] = x;
15     for(int i = 0; i <= step; ++i)
16     {
17         if(dfs(x + a[i], step + 1)) return true;
18         if(dfs(abs(x - a[i]), step + 1)) return true;
19     }
20     return false;
21 }
22
23 int main()
24 {
25     while(scanf("%d", &n) != EOF && n)
26     {
27         if(ans[n]) { printf("%d\n", ans[n]); continue; }
```

```
28         for(deep = 0; ;++deep)
29             if(dfs(1, 0)) {printf("%d\n", ans[n] = deep); break;}
30     }
31     return 0;
32 }
```



## T4.天气预报

- 搜索枚举每一天降雨的云的位置：
  - 判断下雨的位置会不会覆盖赶集或过节的日子。
  - 是否存在有地方连续7天不下雨。
- 优化1：判断连续7天不下雨，不需要判断每一个位置，只需要判断四个角(1, 4, 13, 16)是否连续7天不下雨。
- 优化2：用记忆化记录第 $day$ 天，云在 $(x, y)$ ，四个角连续几天不下雨的是否会失败，在遇到同样状态时直接判断失败。

```
1 // 只需要记录四个角(1, 4, 13, 16)的最近降雨情况即可
2 // 四个角是否合法等价于整个图是否合法
3 #include <bits/stdc++.h>
4 using namespace std;
5 const int N = 370;
6 int d[9][2] = { {0, 0}, {-2, 0}, {-1, 0}, {1, 0}, {2, 0}, {0, -2}, {0, -1}, {0, 1}, {0, 2} };
7 int tb[N][5][5];
8 bool vis[5][5][N][8][8][8][8]; // 记忆化
9 int n;
10
11 // 四个角最近下雨是什么时候
12 struct Node
13 {
14     int a, b, c, d;
15 };
16
17 // 第day天 云在(x, y)
18 bool dfs(int x, int y, int day, Node s)
19 {
20     if(vis[x][y][day][s.a][s.b][s.c][s.d]) return false; // 状态只要访问过 一定是失败的
21     if(day - s.a >= 7 || day - s.b >= 7 || day - s.c >= 7 || day - s.d >= 7)
22     {
23         vis[x][y][day][s.a][s.b][s.c][s.d] = true;
24         return false;
25     }
```

```

25     }
26     if(day == n) return true;
27     for(int i = 0; i <= 8; ++i)
28     {
29         int nx = x + d[i][0], ny = y + d[i][1];
30         if(nx < 1 || nx > 3 || ny < 1 || ny > 3) continue;
31         if(tb[day + 1][nx][ny] || tb[day + 1][nx + 1][ny] || tb[day + 1][nx][ny + 1] || tb[day + 1][nx
+ 1][ny + 1]) continue;
32         Node ts = s;
33         if(nx == 1 && ny == 1) ts.a = day + 1;
34         if(nx == 1 && ny == 3) ts.b = day + 1;
35         if(nx == 3 && ny == 1) ts.c = day + 1;
36         if(nx == 3 && ny == 3) ts.d = day + 1;
37         if(dfs(nx, ny, day + 1, ts)) return true;
38     }
39     vis[x][y][day][s.a][s.b][s.c][s.d] = true;
40     return false;
41 }
42
43 int main()
44 {
45     while(scanf("%d", &n) != EOF && n)
46     {
47         memset(tb, 0, sizeof(tb));
48         memset(vis, 0, sizeof(vis));
49         for(int i = 1; i <= n; ++i)
50             for(int j = 1; j <= 4; ++j)
51                 for(int k = 1; k <= 4; ++k)
52                     scanf("%d", &tb[i][j][k]);
53         if(tb[1][2][2] || tb[1][2][3] || tb[1][3][2] || tb[1][3][3]) { puts("0"); continue; }
54         if(dfs(2, 2, 1, {0, 0, 0, 0})) puts("1");
55         else puts("0");
56     }
57     return 0;
58 }

```

