

二进制枚举

二进制枚举是解决子集枚举一类问题的方法，对于一个包含 n 个元素的集合，其子集的数量为 2^n 种。通过二进制枚举，我们可以高效地遍历所有可能的子集。

蛋糕的美味值: <https://www.lanqiao.cn/problems/8664/learning/>

递归实现指数型枚举: <https://www.lanqiao.cn/problems/19685/learning>

五子棋对弈: <https://www.lanqiao.cn/problems/19694/learning>

基本原理

- 每个子集可以用一个长度为 n 的二进制数表示。
- 二进制数的每一位表示集合中对应元素是否被选中：
 - 1 表示选中该元素。
 - 0 表示不选中该元素。
- 例如，集合 $\{a, b, c\}$ 的子集可以用 3 位二进制数表示：
 - 000：空集。
 - 001： $\{c\}$ 。
 - 010： $\{b\}$ 。
 - 011： $\{b, c\}$ 。
 - 100： $\{a\}$ 。
 - 101： $\{a, c\}$ 。
 - 110： $\{a, b\}$ 。
 - 111： $\{a, b, c\}$ 。

模版

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      int n;
6      cin >> n; // 输入集合的大小 n
7
8      // 遍历所有可能的子集 (2^n 种)
9      for (int i = 0; i < (1 << n); i++) {
10         // 检查每个元素是否被选中
11         for (int j = 0; j < n; j++) {
12             if ((i >> j & 1) == 1) { // 如果 i 的第 j 位为 1
13                 // 这里可以处理选中的元素
14                 // 例如：输出选中的元素
15                 cout << j << " ";
16             }
17         }
18         cout << "\n"; // 换行，表示一个子集结束
19     }
20
21     return 0;
22 }
```

```

1  import java.util.Scanner;
2
3  public class BinaryEnumeration {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          int n = sc.nextInt(); // 输入集合的大小 n
7
8          // 遍历所有可能的子集 (2^n 种)
9          for (int i = 0; i < (1 << n); i++) {
10             // 检查每个元素是否被选中
11             for (int j = 0; j < n; j++) {
12                 if ((i >> j & 1) == 1) { // 如果 i 的第 j 位为 1
13                     // 这里可以处理选中的元素
14                     // 例如: 输出选中的元素
15                     System.out.print(j + " ");
16                 }
17             }
18             System.out.println(); // 换行, 表示一个子集结束
19         }
20     }
21 }

```

```

1  n = int(input()) # 输入集合的大小 n
2
3  # 遍历所有可能的子集 (2^n 种)
4  for i in range(1 << n):
5      # 检查每个元素是否被选中
6      for j in range(n):
7          if (i >> j) & 1 == 1: # 如果 i 的第 j 位为 1
8              # 这里可以处理选中的元素
9              # 例如: 输出选中的元素
10             print(j, end=" ")
11     print() # 换行, 表示一个子集结束

```

五子棋对弈

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  int b[5][5]; // 修正为5x5的棋盘
5  int a[25];   // 修正为25个元素
6
7  bool check() {
8      int pos = 0;
9      for (int i = 0; i < 5; i++) {
10         for (int j = 0; j < 5; j++) {
11             b[i][j] = a[pos++];
12         }
13     }
14
15     // 检查行和列
16     for (int i = 0; i < 5; i++) {
17         int rowSum = 0, colSum = 0;
18         for (int j = 0; j < 5; j++) {
19             rowSum += b[i][j];
20             colSum += b[j][i];

```

```

21     }
22     if (rowSum == 0 || rowSum == 5 || colSum == 0 || colSum == 5) {
23         return false;
24     }
25 }
26
27 // 检查对角线
28 int diag1 = 0, diag2 = 0;
29 for (int i = 0; i < 5; i++) {
30     diag1 += b[i][i];
31     diag2 += b[i][4 - i];
32 }
33 if (diag1 == 0 || diag1 == 5 || diag2 == 0 || diag2 == 5) {
34     return false;
35 }
36
37 return true;
38 }
39
40 int main() {
41     int ans = 0;
42     // 遍历所有可能的25位二进制数
43     for (int i = 0; i < (1 << 25); i++) {
44         int s = 0;
45         for (int j = 0; j < 25; j++) {
46             a[j] = (i >> j) & 1;
47             if (a[j]) {
48                 s++;
49             }
50         }
51         // 只有当棋子数量为13时才进行检查
52         if (s != 13) continue;
53         if (check()) {
54             ans++;
55         }
56     }
57     cout << ans << endl;
58     return 0;
59 }

```

```

1 public class Main {
2     static int[][] b = new int[5][5]; // 5x5 的棋盘
3     static int[] a = new int[25];     // 25 个元素
4
5     // 检查棋盘是否满足条件
6     static boolean check() {
7         int pos = 0;
8         for (int i = 0; i < 5; i++) {
9             for (int j = 0; j < 5; j++) {
10                 b[i][j] = a[pos++];
11             }
12         }
13
14         // 检查行和列
15         for (int i = 0; i < 5; i++) {
16             int rowSum = 0, colSum = 0;
17             for (int j = 0; j < 5; j++) {

```

```

18         rowSum += b[i][j];
19         colSum += b[j][i];
20     }
21     if (rowSum == 0 || rowSum == 5 || colSum == 0 || colSum == 5) {
22         return false;
23     }
24 }
25
26 // 检查对角线
27 int diag1 = 0, diag2 = 0;
28 for (int i = 0; i < 5; i++) {
29     diag1 += b[i][i];
30     diag2 += b[i][4 - i];
31 }
32 if (diag1 == 0 || diag1 == 5 || diag2 == 0 || diag2 == 5) {
33     return false;
34 }
35
36 return true;
37 }
38
39 public static void main(String[] args) {
40     int ans = 0;
41     // 遍历所有可能的 25 位二进制数
42     for (int i = 0; i < (1 << 25); i++) {
43         int s = 0;
44         for (int j = 0; j < 25; j++) {
45             a[j] = (i >> j) & 1;
46             if (a[j] == 1) {
47                 s++;
48             }
49         }
50         // 只有当棋子数量为 13 时才进行检查
51         if (s != 13) continue;
52         if (check()) {
53             ans++;
54         }
55     }
56     System.out.println(ans);
57 }
58 }

```

```

1 def check():
2     global a, b
3     pos = 0
4     for i in range(5):
5         for j in range(5):
6             b[i][j] = a[pos]
7             pos += 1
8
9     # 检查行和列
10    for i in range(5):
11        row_sum = sum(b[i][j] for j in range(5))
12        col_sum = sum(b[j][i] for j in range(5))
13        if row_sum == 0 or row_sum == 5 or col_sum == 0 or col_sum == 5:
14            return False
15

```

```
16     # 检查对角线
17     diag1 = sum(b[i][i] for i in range(5))
18     diag2 = sum(b[i][4 - i] for i in range(5))
19     if diag1 == 0 or diag1 == 5 or diag2 == 0 or diag2 == 5:
20         return False
21
22     return True
23
24 def main():
25     global a, b
26     b = [[0] * 5 for _ in range(5)] # 5x5 的棋盘
27     a = [0] * 25                     # 25 个元素
28     ans = 0
29
30     # 遍历所有可能的 25 位二进制数
31     for i in range(1 << 25):
32         s = 0
33         for j in range(25):
34             a[j] = (i >> j) & 1
35             if a[j] == 1:
36                 s += 1
37         # 只有当棋子数量为 13 时才进行检查
38         if s != 13:
39             continue
40         if check():
41             ans += 1
42
43     print(ans)
44
45 if __name__ == "__main__":
46     main()
```