

递归实现排列型枚举

递归实现排列型枚举: <https://www.lanqiao.cn/problems/19684/learning/>

串变换: <https://www.lanqiao.cn/problems/4360/learning/> 3 星

带分数: <https://www.lanqiao.cn/problems/208/learning/> 3 星

考虑暴力枚举该问题:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int,int> PII;
5  int main(){
6      int n=3;
7      for(int i=1;i<=n;i++){
8          st[i]=true;
9          for(int j=1;j<=n;j++){
10             if(st[j]){
11                 continue;
12             }
13             st[j]=true;
14             for(int k=1;k<=n;k++){
15                 if(st[k]) {
16                     continue;
17                 }
18                 st[k]=true;
19                 printf("%d %d %d\n",i,j,k);
20                 st[k]=false;
21             }
22             st[j]=false;
23         }
24         st[i]=false;
25     }
26 }
27
```

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          int n = 3;
6          boolean[] st = new boolean[n + 1];
7
8          for (int i = 1; i <= n; i++) {
9              st[i] = true;
10             for (int j = 1; j <= n; j++) {
11                 if (st[j]) {
12                     continue;
13                 }
14                 st[j] = true;
15                 for (int k = 1; k <= n; k++) {
16                     if (st[k]) {
17                         continue;
18                     }
19                 }
20                 st[k] = true;
21                 printf("%d %d %d\n", i, j, k);
22                 st[k] = false;
23             }
24             st[j] = false;
25         }
26         st[i] = false;
27     }
28 }
```

```

18         }
19         st[k] = true;
20         System.out.println(i + " " + j + " " + k);
21         st[k] = false;
22     }
23     st[j] = false;
24 }
25 st[i] = false;
26 }
27 }
28 }
29

```

```

1  n = 3
2  st = [False] * (n + 1)
3
4  for i in range(1, n + 1):
5      st[i] = True
6      for j in range(1, n + 1):
7          if st[j]:
8              continue
9          st[j] = True
10         for k in range(1, n + 1):
11             if st[k]:
12                 continue
13             st[k] = True
14             print(i, j, k)
15             st[k] = False
16         st[j] = False
17     st[i] = False
18

```

递归

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int n;
4  int a[10], vis[10];
5  void dfs(int cnt)
6  {
7      if(cnt == n)//到限制了直接输出
8      {
9          for(int i = 1;i <= n;i ++)
10             cout << a[i] << ' ';
11         cout << '\n';
12         return;
13     }
14     for(int i = 1;i <= n;i ++)
15     {
16         if(!vis[i])
17         {
18             vis[i] = 1;
19             a[cnt + 1] = i;//储存以下当前的排列情况再递归
20             dfs(cnt + 1);//递归下一个可能的数
21             //a[cnt + 1] = 0;可有可无的回溯
22             vis[i] = 0;//回溯

```

```

23     }
24 }
25 }
26 int main()
27 {
28     cin >> n;
29     dfs(0);
30     return 0;
31 }

```

```

1  import java.util.Scanner;
2  // 1:无需package
3  // 2: 类名必须Main, 不可修改
4
5  public class Main {
6      // 定义一个布尔类型的数组, 用于标记数字是否已经在当前排列中使用过
7      static boolean[] a;
8      // 定义一个整数变量n, 表示要生成全排列的数字范围是从1到n
9      static int n;
10     // 定义一个整数类型的数组b, 用于存储当前正在生成的排列
11     static int[] b;
12
13     public static void main(String[] args) {
14         Scanner in = new Scanner(System.in);
15         // 读取一个整数, 并赋值给n
16         n = in.nextInt();
17         // 初始化布尔类型数组a, 长度为n+1, 初始值都为false, 表示所有数字都未被使用
18         a = new boolean[n + 1];
19         // 初始化整数数组b, 数组长度为n+1, 用于存储排序
20         b = new int[n + 1];
21         // 调用深度优先搜索函数dfs, 从第1个位置开始生成排序
22         dfs(1);
23     }
24
25     // 定义深度优先搜索函数dfs, x表示当前要填充的位置
26     static void dfs(int x) {
27         // 如果当前要填充的位置x大于n, 表示已经生成了一个完整的排列
28         if (x > n) {
29             // 遍历数组b, 输出当前排列中的每个数字, 数字之间用空格分隔开
30             for (int i = 1; i <= n; i++) {
31                 System.out.print(b[i] + " ");
32             }
33             // 输出换行符, 准备输出下一个排列
34             System.out.println();
35             // 返回上一层递归
36             return;
37         }
38         // 尝试将1到n中的每个数字填入当前位置x
39         for (int i = 1; i <= n; i++) {
40             // 如果数字i还未被使用
41             if (!a[i]) {
42                 // 标记数组i已经被使用
43                 a[i] = true;
44                 // 将数字i填入数组b的第x个位置
45                 b[x] = i;
46                 // 递归调用dfs函数, 填充下一个位置x+1
47                 dfs(x + 1);

```

```

48         // 回溯操作，将数字i标记为未使用，以便尝试其它肯的排列
49         a[i] = false;
50     }
51 }
52 }
53 }

```

```

1 def dfs(x):
2     # 如果当前要填充的位置 x 大于 n，表示已经生成了一个完整的排列
3     if x > n:
4         print(" ".join(map(str, b[1:]))) # 输出当前排列
5         return
6
7     # 尝试将 1 到 n 中的每个数字填入当前位置 x
8     for i in range(1, n + 1):
9         if not a[i]: # 如果数字 i 还未被使用
10            a[i] = True # 标记数字 i 已被使用
11            b[x] = i # 将数字 i 填入数组 b 的第 x 个位置
12            dfs(x + 1) # 递归调用 dfs 填充下一个位置
13            a[i] = False # 回溯，将数字 i 标记为未使用
14
15
16 # 读取输入
17 n = int(input()) # 读取一个整数，并赋值给 n
18 a = [False] * (n + 1) # 初始化布尔数组，表示所有数字未被使用
19 b = [0] * (n + 1) # 初始化整数数组 b，用于存储排列
20
21 dfs(1) # 调用深度优先搜索函数，从第 1 个位置开始生成排列
22

```

N 皇后

<https://www.lanqiao.cn/problems/1508/learning/>

考虑暴力枚举该问题：

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 int row[30], col[30], zhu[30], fu[30];
5
6 int main() {
7     int n = 4;
8     int ans=0;
9     for (int i = 1; i <= n; i++) {
10        int x1 = 1, y1 = i;
11        row[x1] = true;
12        col[y1] = true;
13        fu[x1 + y1] = true;
14        zhu[n - y1 + x1] = true;
15
16        for (int j = 1; j <= n; j++) {
17            int x2 = 2, y2 = j;
18            if (row[x2] || col[y2] || fu[x2 + y2] || zhu[n - y2 + x2])
19                continue;
20            row[x2] = true;
21

```

```

20         col[y2] = true;
21         fu[x2 + y2] = true;
22         zhu[n - y2 + x2] = true;
23
24         for (int k = 1; k <= n; k++) {
25             int x3 = 3, y3 = k;
26             if (row[x3] || col[y3] || fu[x3 + y3] || zhu[n - y3 + x3])
continue;
27             row[x3] = true;
28             col[y3] = true;
29             fu[x3 + y3] = true;
30             zhu[n - y3 + x3] = true;
31
32             for (int l = 1; l <= n; l++) {
33                 int x4 = 4, y4 = l;
34                 if (row[x4] || col[y4] || fu[x4 + y4] || zhu[n - y4 +
x4]) continue;
35                 ans++;
36             }
37
38             // 回溯
39             row[x3] = false;
40             col[y3] = false;
41             fu[x3 + y3] = false;
42             zhu[n - y3 + x3] = false;
43         }
44
45         // 回溯
46         row[x2] = false;
47         col[y2] = false;
48         fu[x2 + y2] = false;
49         zhu[n - y2 + x2] = false;
50     }
51
52     // 回溯
53     row[x1] = false;
54     col[y1] = false;
55     fu[x1 + y1] = false;
56     zhu[n - y1 + x1] = false;
57 }
58 return 0;
59 }

```

```

1 public class Main {
2     static int[] row = new int[30];
3     static int[] col = new int[30];
4     static int[] zhu = new int[30];
5     static int[] fu = new int[30];
6
7     public static void main(String[] args) {
8         int n = 4;
9         int ans = 0;
10
11         for (int i = 1; i <= n; i++) {
12             int x1 = 1, y1 = i;
13             row[x1] = 1;
14             col[y1] = 1;

```

```

15         fu[x1 + y1] = 1;
16         zhu[n - y1 + x1] = 1;
17
18         for (int j = 1; j <= n; j++) {
19             int x2 = 2, y2 = j;
20             if (row[x2] == 1 || col[y2] == 1 || fu[x2 + y2] == 1 ||
zhu[n - y2 + x2] == 1) continue;
21             row[x2] = 1;
22             col[y2] = 1;
23             fu[x2 + y2] = 1;
24             zhu[n - y2 + x2] = 1;
25
26             for (int k = 1; k <= n; k++) {
27                 int x3 = 3, y3 = k;
28                 if (row[x3] == 1 || col[y3] == 1 || fu[x3 + y3] == 1 ||
zhu[n - y3 + x3] == 1) continue;
29                 row[x3] = 1;
30                 col[y3] = 1;
31                 fu[x3 + y3] = 1;
32                 zhu[n - y3 + x3] = 1;
33
34                 for (int l = 1; l <= n; l++) {
35                     int x4 = 4, y4 = l;
36                     if (row[x4] == 1 || col[y4] == 1 || fu[x4 + y4] ==
1 || zhu[n - y4 + x4] == 1) continue;
37                     ans++;
38                 }
39
40                 // 回溯
41                 row[x3] = 0;
42                 col[y3] = 0;
43                 fu[x3 + y3] = 0;
44                 zhu[n - y3 + x3] = 0;
45             }
46
47             // 回溯
48             row[x2] = 0;
49             col[y2] = 0;
50             fu[x2 + y2] = 0;
51             zhu[n - y2 + x2] = 0;
52         }
53
54         // 回溯
55         row[x1] = 0;
56         col[y1] = 0;
57         fu[x1 + y1] = 0;
58         zhu[n - y1 + x1] = 0;
59     }
60
61     System.out.println(ans);
62 }
63 }

```

```

1  def main():
2      n = 4
3      row = [0] * 30
4      col = [0] * 30

```

```

5     zhu = [0] * 30
6     fu = [0] * 30
7     ans = 0
8
9     for i in range(1, n + 1):
10         x1, y1 = 1, i
11         row[x1] = 1
12         col[y1] = 1
13         fu[x1 + y1] = 1
14         zhu[n - y1 + x1] = 1
15
16         for j in range(1, n + 1):
17             x2, y2 = 2, j
18             if row[x2] or col[y2] or fu[x2 + y2] or zhu[n - y2 + x2]:
19                 continue
20             row[x2] = 1
21             col[y2] = 1
22             fu[x2 + y2] = 1
23             zhu[n - y2 + x2] = 1
24
25             for k in range(1, n + 1):
26                 x3, y3 = 3, k
27                 if row[x3] or col[y3] or fu[x3 + y3] or zhu[n - y3 + x3]:
28                     continue
29                 row[x3] = 1
30                 col[y3] = 1
31                 fu[x3 + y3] = 1
32                 zhu[n - y3 + x3] = 1
33
34                 for l in range(1, n + 1):
35                     x4, y4 = 4, l
36                     if row[x4] or col[y4] or fu[x4 + y4] or zhu[n - y4 +
x4]:
37                         continue
38                     ans += 1
39
40                     # 回溯
41                     row[x3] = 0
42                     col[y3] = 0
43                     fu[x3 + y3] = 0
44                     zhu[n - y3 + x3] = 0
45
46                     # 回溯
47                     row[x2] = 0
48                     col[y2] = 0
49                     fu[x2 + y2] = 0
50                     zhu[n - y2 + x2] = 0
51
52                     # 回溯
53                     row[x1] = 0
54                     col[y1] = 0
55                     fu[x1 + y1] = 0
56                     zhu[n - y1 + x1] = 0
57
58     print(ans)
59
60 if __name__ == "__main__":
61     main()

```

dfs

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int,int> PII;
5  //按行枚举
6  int ans=0;
7  int n;
8  const int N=30;
9  bool col[N],fan[N],zhu[N];
10 void dfs(int u){//看每一行是否有位置
11 //可以填棋子，如果能填完最后一行，就是一个可行解
12     if(u>n){
13         ans++;
14         return;
15     }
16     for(int i=1;i<=n;i++){
17         //看每一行每一列是否能填写棋子
18         if(!col[i]&&!fan[u+i]&&!zhu[n-i+u]){
19             col[i]=true;
20             fan[u+i]=true;
21             zhu[n-i+u]=true;
22             dfs(u+1);
23             zhu[n-i+u]=false;
24             fan[u+i]=false;
25             col[i]=false;
26         }
27     }
28 }
29 void solve(){
30     cin>>n;
31     dfs(1);
32     cout<<ans;
33 }
34 int main(){
35     // ios::sync_with_stdio(false);cin.tie(0);
36     int t=1;
37     // scanf("%d",&t);
38     // cin>>t;
39     while(t--) solve();
40 }
```

```
1  import java.util.Scanner;
2
3  public class NQueens {
4      static int ans = 0;
5      static int n;
6      static boolean[] col = new boolean[30];
7      static boolean[] fan = new boolean[30];
8      static boolean[] zhu = new boolean[30];
9
10     public static void dfs(int u) {
11         if (u > n) {
12             ans++;
13         }
14     }
15 }
```



```

13         return;
14     }
15     for (int i = 1; i <= n; i++) {
16         if (!col[i] && !fan[u + i] && !zhu[n - i + u]) {
17             col[i] = fan[u + i] = zhu[n - i + u] = true;
18             dfs(u + 1);
19             col[i] = fan[u + i] = zhu[n - i + u] = false;
20         }
21     }
22 }
23
24 public static void solve() {
25     Scanner scanner = new Scanner(System.in);
26     n = scanner.nextInt();
27     ans = 0;
28     dfs(1);
29     System.out.println(ans);
30     scanner.close();
31 }
32
33 public static void main(String[] args) {
34     solve();
35 }
36 }

```

```

1 def dfs(u, n, col, fan, zhu):
2     global ans
3     if u > n:
4         ans += 1
5         return
6     for i in range(1, n + 1):
7         if not col[i] and not fan[u + i] and not zhu[n - i + u]:
8             col[i] = fan[u + i] = zhu[n - i + u] = True
9             dfs(u + 1, n, col, fan, zhu)
10            col[i] = fan[u + i] = zhu[n - i + u] = False
11
12 def solve():
13     global ans
14     n = int(input())
15     ans = 0
16     col = [False] * 30
17     fan = [False] * 30
18     zhu = [False] * 30
19     dfs(1, n, col, fan, zhu)
20     print(ans)
21
22 if __name__ == "__main__":
23     solve()
24

```

子集枚举（递归实现指数型枚举）

递归实现指数型枚举: <https://www.lanqiao.cn/problems/19685/learning/> 2星

递归实现组合型枚举: <https://www.lanqiao.cn/problems/19880/learning/> 3星

蛋糕的美味值: <https://www.lanqiao.cn/problems/8664/learning> 2星

暴力

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int,int> PII;
5  const int N=30;
6  int n;
7  int st[10];
8  int main(){
9      int n=3;
10     for(int i=0;i<=1;i++){
11         st[1]=i;
12         for(int j=0;j<=1;j++){
13             st[2]=j;
14             for(int k=0;k<=1;k++){
15                 st[3]=k;
16                 for(int l=1;l<=n;l++){
17                     if(st[l]) printf("%d ",l);
18                 }
19                 printf("\n");
20             }
21         }
22     }
23 }
24

```

```

1  public class Main {
2      public static void main(String[] args) {
3          int n = 3;
4          int[] st = new int[10];
5
6          for (int i = 0; i <= 1; i++) {
7              st[1] = i;
8              for (int j = 0; j <= 1; j++) {
9                  st[2] = j;
10                 for (int k = 0; k <= 1; k++) {
11                     st[3] = k;
12                     for (int l = 1; l <= n; l++) {
13                         if (st[l] == 1) {
14                             System.out.print(l + " ");
15                         }
16                     }
17                     System.out.println();
18                 }
19             }
20         }
21     }
22 }
23

```

```

1  def main():
2      n = 3
3      st = [0] * 10

```

```

4     for i in range(2):
5         st[1] = i
6         for j in range(2):
7             st[2] = j
8             for k in range(2):
9                 st[3] = k
10                print(" ".join(str(l) for l in range(1, n + 1) if st[l]))
11
12 if __name__ == "__main__":
13     main()
14

```

dfs

```

1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  const int N = 16;
6
7  int n;
8  bool st[N];
9
10 void dfs(int u)
11 {
12     if(u > n)
13     {
14         for(int i = 1; i <= n; i++)
15         {
16             if(st[i]) cout << i << " ";
17         }
18         cout << endl;
19         return;
20     }
21     st[u] = false;
22     dfs(u + 1);
23     st[u] = true;
24     dfs(u + 1);
25 }
26
27 int main()
28 {
29     cin >> n;
30
31     dfs(1);
32     return 0;
33 }

```

```

1  import java.util.Scanner;
2
3  public class Main {
4      static int n;
5      static boolean[] st = new boolean[16];
6
7      public static void dfs(int u) {
8          if (u > n) {

```

```

9         for (int i = 1; i <= n; i++) {
10             if (st[i]) System.out.print(i + " ");
11         }
12         System.out.println();
13         return;
14     }
15     st[u] = false;
16     dfs(u + 1);
17     st[u] = true;
18     dfs(u + 1);
19 }
20
21 public static void main(String[] args) {
22     Scanner scanner = new Scanner(System.in);
23     n = scanner.nextInt();
24     scanner.close();
25     dfs(1);
26 }
27 }
28

```

```

1  def dfs(u, n, st):
2      if u > n:
3          print(" ".join(str(i) for i in range(1, n + 1) if st[i]))
4          return
5      st[u] = False
6      dfs(u + 1, n, st)
7      st[u] = True
8      dfs(u + 1, n, st)
9
10 def main():
11     n = int(input())
12     st = [False] * (n + 1)
13     dfs(1, n, st)
14
15 if __name__ == "__main__":
16     main()
17

```