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：

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
int main(){
    int n=3;
    for(int i=1;i<=n;i++){
        st[i]=true;
        for(int j=1;j<=n;j++){
            if(st[j]){
                 continue;
            }
            st[j]=true;
            for(int k=1;k<=n;k++){
                if(st[k]) {
                    continue;
                }
                st[k]=true;
                printf("%d %d %d\n",i,j,k);
                st[k]=false;
            }
            st[j]=false;
        }
        st[i]=false;
    }
}

```

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        int n = 3;
        boolean[] st = new boolean[n + 1];

        for (int i = 1; i <= n; i++) {
            st[i] = true;
            for (int j = 1; j <= n; j++) {
                if (st[j]) {
                    continue;
                }
                st[j] = true;
                for (int k = 1; k <= n; k++) {
                    if (st[k]) {
                        continue;
                    }
                    st[k] = true;
                    System.out.println(i + " " + j + " " + k);
                    st[k] = false;
                }
                st[j] = false;
            }
            st[i] = false;
        }
    }
}

```

```
n = 3
st = [False] * (n + 1)

for i in range(1, n + 1):
    st[i] = True
    for j in range(1, n + 1):
        if st[j]:
            continue
        st[j] = True
        for k in range(1, n + 1):
            if st[k]:
                continue
            st[k] = True
            print(i, j, k)
            st[k] = False
        st[j] = False
    st[i] = False

```

递归

```
#include <bits/stdc++.h>
using namespace std;
int n;
int a[10], vis[10];
void dfs(int cnt)
{
if(cnt == n)//到限制了直接输出
{
  for(int i = 1;i <= n;i ++)
    cout << a[i] << ' ';
  cout << '\n';
  return;
}
for(int i = 1;i <=n;i ++)
{
  if(!vis[i])
  {
    vis[i] = 1;
    a[cnt + 1] = i;//储存以下当前的排列情况再递归
    dfs(cnt + 1);//递归下一个可能的数
    //a[cnt + 1] = 0;可有可无的回溯
    vis[i] = 0;//回溯
  }
}
}
int main()
{
cin >> n;
dfs(0);
return 0;
}
```

```
import java.util.Scanner;
// 1:无需package
// 2: 类名必须Main, 不可修改

public class Main {
    // 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
    static boolean[] a;
    // 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
    static int n;
    // 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
    static int[] b;

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 读取一个整数，并赋值给n
        n = in.nextInt();
        // 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
        a = new boolean[n + 1];
        // 初始化整数数组b，数组长度为n+1，用于存储排序
        b = new int[n + 1];
        // 调用深度优先搜索函数dfs，从第1个位置开始生成排序
        dfs(1);
    }

    // 定义深度优先搜索函数dfs，x表示当前要填充的位置
    static void dfs(int x) {
        // 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
        if (x > n) {
            // 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
            for (int i = 1; i <= n; i++) {
                System.out.print(b[i] + " ");
            }
            // 输出换行符，准备输出下一个排列
            System.out.println();
            // 返回上一层递归
            return;
        }
        // 尝试将1到n中的每个数字填入当前位置x
        for (int i = 1; i <= n; i++) {
            // 如果数字i还未被使用
            if (!a[i]) {
                // 标记数组i已经被使用
                a[i] = true;
                // 将数字i填入数组b的第x个位置
                b[x] = i;
                // 递归调用dfs函数，填充下一个位置x+1
                dfs(x + 1);
                // 回溯操作，将数字i标记为未使用，以便尝试其它肯的排列
                a[i] = false;
            }
        }
    }
}
```

```python
def dfs(x):
    # 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
    if x > n:
        print(" ".join(map(str, b[1:])))  # 输出当前排列
        return

    # 尝试将 1 到 n 中的每个数字填入当前位置 x
    for i in range(1, n + 1):
        if not a[i]:  # 如果数字 i 还未被使用
            a[i] = True  # 标记数字 i 已被使用
            b[x] = i  # 将数字 i 填入数组 b 的第 x 个位置
            dfs(x + 1)  # 递归调用 dfs 填充下一个位置
            a[i] = False  # 回溯，将数字 i 标记为未使用


# 读取输入
n = int(input())  # 读取一个整数，并赋值给 n
a = [False] * (n + 1)  # 初始化布尔数组，表示所有数字未被使用
b = [0] * (n + 1)  # 初始化整数数组 b，用于存储排列

dfs(1)  # 调用深度优先搜索函数，从第 1 个位置开始生成排列

```

N 皇后

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

考虑暴力枚举该问题：

```cpp
#include<bits/stdc++.h>
using namespace std;

int row[30], col[30], zhu[30], fu[30];

int main() {
    int n = 4;
    int ans=0;
    for (int i = 1; i <= n; i++) {
        int x1 = 1, y1 = i;
        row[x1] = true;
        col[y1] = true;
        fu[x1 + y1] = true;
        zhu[n - y1 + x1] = true;

        for (int j = 1; j <= n; j++) {
            int x2 = 2, y2 = j;
            if (row[x2] || col[y2] || fu[x2 + y2] || zhu[n - y2 + x2]) 
continue;
            row[x2] = true;
```

```java
            col[y2] = true;
            fu[x2 + y2] = true;
            zhu[n - y2 + x2] = true;

            for (int k = 1; k <= n; k++) {
                int x3 = 3, y3 = k;
                if (row[x3] || col[y3] || fu[x3 + y3] || zhu[n - y3 + x3]) continue;
                row[x3] = true;
                col[y3] = true;
                fu[x3 + y3] = true;
                zhu[n - y3 + x3] = true;

                for (int l = 1; l <= n; l++) {
                    int x4 = 4, y4 = l;
                    if (row[x4] || col[y4] || fu[x4 + y4] || zhu[n - y4 + x4]) continue;
                    ans++;
                }

                // 回溯
                row[x3] = false;
                col[y3] = false;
                fu[x3 + y3] = false;
                zhu[n - y3 + x3] = false;
            }

            // 回溯
            row[x2] = false;
            col[y2] = false;
            fu[x2 + y2] = false;
            zhu[n - y2 + x2] = false;
        }

        // 回溯
        row[x1] = false;
        col[y1] = false;
        fu[x1 + y1] = false;
        zhu[n - y1 + x1] = false;
    }
    return 0;
}
```

```java
public class Main {
    static int[] row = new int[30];
    static int[] col = new int[30];
    static int[] zhu = new int[30];
    static int[] fu = new int[30];

    public static void main(String[] args) {
        int n = 4;
        int ans = 0;

        for (int i = 1; i <= n; i++) {
            int x1 = 1, y1 = i;
            row[x1] = 1;
            col[y1] = 1;
            fu[x1 + y1] = 1;
            zhu[n - y1 + x1] = 1;

            for (int j = 1; j <= n; j++) {
                int x2 = 2, y2 = j;
                if (row[x2] == 1 || col[y2] == 1 || fu[x2 + y2] == 1 || zhu[n - y2 + x2] == 1) continue;
                row[x2] = 1;
                col[y2] = 1;
                fu[x2 + y2] = 1;
                zhu[n - y2 + x2] = 1;

                for (int k = 1; k <= n; k++) {
                    int x3 = 3, y3 = k;
                    if (row[x3] == 1 || col[y3] == 1 || fu[x3 + y3] == 1 || zhu[n - y3 + x3] == 1) continue;
                    row[x3] = 1;
                    col[y3] = 1;
                    fu[x3 + y3] = 1;
                    zhu[n - y3 + x3] = 1;

                    for (int l = 1; l <= n; l++) {
                        int x4 = 4, y4 = l;
                        if (row[x4] == 1 || col[y4] == 1 || fu[x4 + y4] == 1 || zhu[n - y4 + x4] == 1) continue;
                        ans++;
                    }

                    // 回溯
                    row[x3] = 0;
                    col[y3] = 0;
                    fu[x3 + y3] = 0;
                    zhu[n - y3 + x3] = 0;
                }

                // 回溯
                row[x2] = 0;
                col[y2] = 0;
                fu[x2 + y2] = 0;
                zhu[n - y2 + x2] = 0;
            }

            // 回溯
            row[x1] = 0;
            col[y1] = 0;
            fu[x1 + y1] = 0;
            zhu[n - y1 + x1] = 0;
        }

        System.out.println(ans);
    }
}
```

```python
def main():
    n = 4
    row = [0] * 30
    col = [0] * 30
    zhu = [0] * 30
    fu = [0] * 30
    ans = 0

    for i in range(1, n + 1):
        x1, y1 = 1, i
        row[x1] = 1
        col[y1] = 1
        fu[x1 + y1] = 1
        zhu[n - y1 + x1] = 1

        for j in range(1, n + 1):
            x2, y2 = 2, j
            if row[x2] or col[y2] or fu[x2 + y2] or zhu[n - y2 + x2]:
                continue
            row[x2] = 1
            col[y2] = 1
            fu[x2 + y2] = 1
            zhu[n - y2 + x2] = 1

            for k in range(1, n + 1):
                x3, y3 = 3, k
                if row[x3] or col[y3] or fu[x3 + y3] or zhu[n - y3 + x3]:
                    continue
                row[x3] = 1
                col[y3] = 1
                fu[x3 + y3] = 1
                zhu[n - y3 + x3] = 1

                for l in range(1, n + 1):
                    x4, y4 = 4, l
                    if row[x4] or col[y4] or fu[x4 + y4] or zhu[n - y4 + x4]:
                        continue
                    ans += 1

                # 回溯
                row[x3] = 0
                col[y3] = 0
                fu[x3 + y3] = 0
                zhu[n - y3 + x3] = 0

            # 回溯
            row[x2] = 0
            col[y2] = 0
            fu[x2 + y2] = 0
            zhu[n - y2 + x2] = 0

        # 回溯
        row[x1] = 0
        col[y1] = 0
        fu[x1 + y1] = 0
        zhu[n - y1 + x1] = 0

    print(ans)

if __name__ == "__main__":
    main()
```

dfs

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
//按行枚举
int ans=0;
int n;
const int N=30;
bool col[N],fan[N],zhu[N];
void dfs(int u){//看每一行是否有位置
//可以填棋子，如果能填完最后一行，就是一个可行解
    if(u>n){
        ans++;
        return;
    }
    for(int i=1;i<=n;i++){
        //看每一行每一列是否能填写棋子
        if(!col[i]&&!fan[u+i]&&!zhu[n-i+u]){
            col[i]=true;
            fan[u+i]=true;
            zhu[n-i+u]=true;
            dfs(u+1);
            zhu[n-i+u]=false;
            fan[u+i]=false;
            col[i]=false;
        }
    }
}
void solve(){
    cin>>n;
    dfs(1);
    cout<<ans;
}
int main(){
    // ios::sync_with_stdio(false);cin.tie(0);
    int t=1;
    // scanf("%d",&t);
    // cin>>t;
    while(t--) solve();
}
```

```java
import java.util.Scanner;

public class NQueens {
    static int ans = 0;
    static int n;
    static boolean[] col = new boolean[30];
    static boolean[] fan = new boolean[30];
    static boolean[] zhu = new boolean[30];

    public static void dfs(int u) {
        if (u > n) {
            ans++;
```

```java
                return;
            }
            for (int i = 1; i <= n; i++) {
                if (!col[i] && !fan[u + i] && !zhu[n - i + u]) {
                    col[i] = fan[u + i] = zhu[n - i + u] = true;
                    dfs(u + 1);
                    col[i] = fan[u + i] = zhu[n - i + u] = false;
                }
            }
        }

    public static void solve() {
        Scanner scanner = new Scanner(System.in);
        n = scanner.nextInt();
        ans = 0;
        dfs(1);
        System.out.println(ans);
        scanner.close();
    }

    public static void main(String[] args) {
        solve();
    }
}
```

```python
def dfs(u, n, col, fan, zhu):
    global ans
    if u > n:
        ans += 1
        return
    for i in range(1, n + 1):
        if not col[i] and not fan[u + i] and not zhu[n - i + u]:
            col[i] = fan[u + i] = zhu[n - i + u] = True
            dfs(u + 1, n, col, fan, zhu)
            col[i] = fan[u + i] = zhu[n - i + u] = False

def solve():
    global ans
    n = int(input())
    ans = 0
    col = [False] * 30
    fan = [False] * 30
    zhu = [False] * 30
    dfs(1, n, col, fan, zhu)
    print(ans)

if __name__ == "__main__":
    solve()

```

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

选数 https://www.luogu.com.cn/problem/P1036 3星

暴力

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
const int N=30;
int n;
int st[10];
int main(){
    int n=3;
    for(int i=0;i<=1;i++){
        st[1]=i;
        for(int j=0;j<=1;j++){
            st[2]=j;
            for(int k=0;k<=1;k++){
                st[3]=k;
                for(int l=1;l<=n;l++){
                    if(st[l]) printf("%d ",l);
                }
                printf("\n");
            }
        }
    }
}

```

```java
public class Main {
    public static void main(String[] args) {
        int n = 3;
        int[] st = new int[10];

        for (int i = 0; i <= 1; i++) {
            st[1] = i;
            for (int j = 0; j <= 1; j++) {
                st[2] = j;
                for (int k = 0; k <= 1; k++) {
                    st[3] = k;
                    for (int l = 1; l <= n; l++) {
                        if (st[l] == 1) {
                            System.out.print(l + " ");
                        }
                    }
                    System.out.println();
                }
            }
        }
    }
}

```

```python
def main():
    n = 3
    st = [0] * 10
    for i in range(2):
        st[1] = i
        for j in range(2):
            st[2] = j
            for k in range(2):
                st[3] = k
                print(" ".join(str(l) for l in range(1, n + 1) if st[l]))

if __name__ == "__main__":
    main()

```

dfs

```cpp
#include<bits/stdc++.h>

using namespace std;

const int N = 16;

int n;
bool st[N];

void dfs(int u)
{
    if(u > n)
    {
        for(int i = 1; i <= n; i ++)
        {
            if(st[i]) cout << i << " ";
        }
        cout << endl;
        return;
    }
    st[u] = false;
    dfs(u + 1);
    st[u] = true;
    dfs(u + 1);
}

int main()
{
    cin >> n;

    dfs(1);
    return 0;
}
```

```java
import java.util.Scanner;

public class Main {
    static int n;
    static boolean[] st = new boolean[16];

    public static void dfs(int u) {
        if (u > n) {
            for (int i = 1; i <= n; i++) {
                if (st[i]) System.out.print(i + " ");
            }
            System.out.println();
            return;
        }
        st[u] = false;
        dfs(u + 1);
        st[u] = true;
        dfs(u + 1);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        n = scanner.nextInt();
        scanner.close();
        dfs(1);
    }
}

```

```python
def dfs(u, n, st):
    if u > n:
        print(" ".join(str(i) for i in range(1, n + 1) if st[i]))
        return
    st[u] = False
    dfs(u + 1, n, st)
    st[u] = True
    dfs(u + 1, n, st)

def main():
    n = int(input())
    st = [False] * (n + 1)
    dfs(1, n, st)

if __name__ == "__main__":
    main()

```