搜索-DFS之搜索顺序

一、AcWing 1116. 马走日

【题目描述】

马在中国象棋以日字形规则移动。

请编写一段程序，给定$n*m$大小的棋盘，以及马的初始位置(x,y)，要求不能重复经过棋盘上的同一个点，计算马可以有多少途径遍历棋盘上的所有点。

【输入格式】

第一行为整数T，表示测试数据组数。

每一组测试数据包含一行，为四个整数，分别为棋盘的大小以及初始位置坐标n,m,x,y。

【输出格式】

每组测试数据包含一行，为一个整数，表示马能遍历棋盘的途径总数，若无法遍历棋盘上的所有点则输出0。

【数据范围】

$1 \le T \le 9$

$1 \le m,n \le 9$

$0 \le x \le n-1$

$0 \le y \le m-1$

【输入样例】

```
1
5 4 0 0
```

【输出样例】

```
32
```

【分析】

直接爆搜马的八个移动方向即可。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 10;
bool st[N][N];
int n, m, x, y, res;
int dx[8] = { -2, -1, 1, 2, 2, 1, -1, -2 };
int dy[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };

void dfs(int x, int y, int step)
{
    if (step == n * m) { res++; return; }
    st[x][y] = true;
    for (int i = 0; i < 8; i++)
    {
        int nx = x + dx[i], ny = y + dy[i];
        if (nx >= 0 && nx < n && ny >= 0 && ny < m && !st[nx][ny])
            dfs(nx, ny, step + 1);
    }
    st[x][y] = false;
}

int main()
{
    int T;
    cin >> T;
    while (T--)
    {
        cin >> n >> m >> x >> y;
        res = 0;
        dfs(x, y, 1);
        cout << res << endl;
    }
    return 0;
}
```

二、AcWing 1117. 单词接龙

【题目描述】

单词接龙是一个与我们经常玩的成语接龙相类似的游戏。

现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”，每个单词最多被使用两次。

在两个单词相连时，其重合部分合为一部分，例如`beast`和`astonish`，如果接成一条龙则变为`beastonish`。

我们可以任意选择重合部分的长度，但其长度必须大于等于1，且严格小于两个串的长度，例如`at`和`atide`间不能相连。

【输入格式】

输入的第一行为一个单独的整数n表示单词数，以下n行每行有一个单词（只含有大写或小写字母，长度不超过20），输入的最后一行为一个单个字符，表示“龙”开头的字母。

你可以假定以此字母开头的“龙”一定存在。

【输出格式】

只需输出以此字母开头的最长的“龙”的长度。

【数据范围】

$n \leq 20$

【输入样例】

```
5
at
touch
cheat
choose
tact
a
```

【输出样例】

```
23
```

【提示】

连成的“龙”为`atoucheatactactouchoose`。

【分析】

如果要使两个可以拼接的字符串拼在一起后长度最长，那么重合的部分应该最短，因此可以从长度为1的子串开始从小到大枚举前一个字符串i的后缀子串以及后一个字符串j的前缀子串，如果两个子串相等长度为k，则记录$g[i][j]=k$，如果$g[i][j]==0$，说明两个字符串无法拼接。最后以所有首字符为题中所给的开头字符的字符串为起点，进行DFS求出可以拼接出的字符串的最大长度即可。

【代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <string>
using namespace std;

const int N = 25;
string str[N];
char start;//开始的首字符
int g[N][N];//g[i][j]表示第i个串与第j个串相接时重合部分的最短长度,为0表示无法相接
int used[N];//used[i]表示第i个串使用的次数
int n, res;

//当前接龙的字符串为s,上一个使用到的串为第last个串
void dfs(string s, int last)
{
    res = max(res, (int)s.size());//size()函数的返回值类型为size_t
    used[last]++;
    for (int i = 0; i < n; i++)
        if (g[last][i] && used[i] < 2)
            dfs(s + str[i].substr(g[last][i]), i);
    used[last]--;
}

int main()
{
    cin >> n;
    for (int i = 0; i < n; i++) cin >> str[i];
    cin >> start;
    //初始化g数组,假设第i个串在前,第j个串在后
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)

        {
```

```
            string a = str[i], b = str[j];
            for (int k = 1; k < min(a.size(), b.size()); k++)
                if (a.substr(a.size() - k, k) == b.substr(0, k))
                {
                    g[i][j] = k;
                    break;
                }
        }
    for (int i = 0; i < n; i++)
        if (str[i][0] == start) dfs(str[i], i);
    cout << res << endl;
    return 0;
}
```

三、AcWing 1118. 分成互质组

【题目描述】

给定n个正整数，将它们分组，使得每组中任意两个数互质。

至少要分成多少个组？

【输入格式】

第一行是一个正整数n。

第二行是n个不大于10000的正整数。

【输出格式】

一个正整数，即最少需要的组数。

【数据范围】

$1 \leq n \leq 10$

【输入样例】

```
6
14 20 33 117 143 175
```

【输出样例】

```
3
```

【分析】

具体思路详见代码注释部分。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 10;
int a[N];
int g[N][N], st[N];
int n, res = N;//最多可以分成10组

int gcd(int a, int b)
{
    return b ? gcd(b, a % b) : a;
}

//判断g中的每个元素是否都和x互质
bool check(int g[], int glen, int x)
{
    for (int i = 0; i < glen; i++)
        if (gcd(g[i], x) != 1) return false;
    return true;
}

//当前是第几组,当前组的长度,从哪个地方开始往后搜,当前总共安排的元素数量
void dfs(int gcnt, int glen, int start, int sum)
{
    if (gcnt >= res) return;//剪枝+防止死循环
    if (sum == n) { res = gcnt; return; }
    bool flag = true;//start之后是否没有任何元素可以放入当前组
    for (int i = start; i < n; i++)
        if (!st[i] && check(g[gcnt], glen, a[i]))
        {
            st[i] = true;
            g[gcnt][glen] = a[i];
            dfs(gcnt, glen + 1, i + 1, sum + 1);
            st[i] = false;
            flag = false;
        }
```

```
    //当所有元素都不能放进当前组或者当start=n-1了但是元素没有全部分组完毕时,要新开一个分组并重新从start=0开始找,并且一定要有st数组,不然会把一个元素重复分组
    if (flag) dfs(gcnt + 1, 0, 0, sum);
}

int main()
{
    cin >> n;
    for (int i = 0; i < n; i++) cin >> a[i];
    dfs(1, 0, 0, 0);
    cout << res << endl;
    return 0;
}
```