DFS剪枝习题题解

A.数的划分

课堂例题，代码课上已经给出，略

B.function

课堂例题，给出了核心代码(~~但QYC估计有同学改不对~~)，现给出AC代码：

```cpp
#include <bits/stdc++.h>
using namespace std;
long long f[21][21][21], INF = 1e9;
long long w(long long a, long long b, long long c)
{
    if (a < 1 || b < 1 || c < 1)
        return 1;
    if (a > 20 || b > 20 || c > 20) {
        if (f[20][20][20] == -INF)
            f[20][20][20] = w(20, 20, 20);
        return f[20][20][20];
    }
    if (a < b && b < c) {
        if (f[a][b][c - 1] == -INF)
            f[a][b][c - 1] = w(a, b, c - 1);
        if (f[a][b - 1][c - 1] == -INF)
            f[a][b - 1][c - 1] = w(a, b - 1, c - 1);
        if (f[a][b - 1][c] == -INF)
            f[a][b - 1][c] = w(a, b - 1, c);
        return f[a][b][c - 1] + f[a][b - 1][c - 1] - f[a][b - 1][c];
    }
    if (f[a - 1][b][c] == -INF)
        f[a - 1][b][c] = w(a - 1, b, c);
    if (f[a - 1][b - 1][c] == -INF)
        f[a - 1][b - 1][c] = w(a - 1, b - 1, c);
    if (f[a - 1][b][c - 1] == -INF)
        f[a - 1][b][c - 1] = w(a - 1, b, c - 1);
    if (f[a - 1][b - 1][c - 1] == -INF)
        f[a - 1][b - 1][c - 1] = w(a - 1, b - 1, c - 1);
    return f[a - 1][b][c] + f[a - 1][b - 1][c] + f[a - 1][b][c - 1] - f[a - 1][b - 1][c - 1];
}
int main()
{
    long long a, b, c;
    for (int i = 0; i < 21; i++)
        for (int j = 0; j < 21; j++)
            for (int k = 0; k < 21; k++)
                f[i][j][k] = -INF;
    cin >> a >> b >> c;
    while (a != -1 || b != -1 || c != -1) {
        cout << "w(" << a << ", " << b << ",  " << c << ") = " << w(a, b, c) << endl;
```

```
        cin >> a >> b >> c;
    }
    return 0;
}
```

C.逃离迷宫

课堂例题，核心代码与正常的网格类题目搜索一致，加上奇偶性剪枝即可通过。核心代码如下：

```
void dfs(int x, int y, int t)  //当前坐标x，y；已经移动的步数t
{
    //边界条件略
    vis[x][y] = true;
    for (int i = 0; i < 4; ++i) {
        int tx = x + dx[i];  //坐标增量
        int ty = y + dy[i];
        if (满足搜索条件) {
            dfs(tx, ty, t + 1);
        }

    }
    vis[x][y] = false;
}
int main()
{
    //输入略
    //奇偶性剪枝
    if ((sx + sy + ex + ey + T) % 2 != 0) {
        cout << "NO" << endl;
    } else {
      //搜索
    }
    return 0;
}
```

D.数字三角形

一道比较经典，多解的问题，除搜索还有其他解法。

主要搜索两个方向，搜索函数$dfs(x,y,sum)$，x表示行，y表示列，sum表示当前的和。

往左下搜索：$dfs(x+1,y,sum+a[x+1][y])$;

往右下搜索：$dfs(x+1,y+1,sum+a[x+1][y+1])$;

边界条件：$x>n$

核心代码：

```
void dfs(int x, int y, int sum) //x表示行，y表示列，sum表示当前的和
{
    //剪枝与边界条件略
    dfs(x + 1, y ,sum + a[x + 1][y]);  //两个方向搜索
    dfs(x + 1, y + 1 ,sum + a[x + 1][y + 1]);
}
```

E.滑雪

按题目要求对四个方向进行搜索，移动时对移动的步数用$f[i][j]$进行记录，保存最优解，体现一个记忆化搜索的思想。

参考代码：

```
int dfs(int x, int y)
{
    //边界条件略
    if (f[x][y] == -INF) { //记忆化，没有更新过就搜索
        for (int i = 1; i <= 4; i++) {
            int dx = x + X[i];
            int dy = y + Y[i];
            if (map[dx][dy] < map[x][y]) {  //记录当前最优的步数，记忆化
                f[x][y] = max(f[x][y], dfs(dx, dy) + 1);
            }
        }
    }
    return f[x][y]; //更新过就直接返回答案
}
```

F.棋盘

这道题目还是比较有意思的，在我们熟悉的棋盘搜索的基础上，加上一个可以使用魔法这样一个条件。

题目如果没有魔法可以使用，那么我们的搜索函数dfs就是这样：$dfs(x,y,nowcoin)$，表示当前的位置(x,y)，以及当前花费的金币

有了魔法，我们的状态就需要知道是否使用魔法，所以需要再加入一个参数$mflag$，表示是否使用魔法。

最终的函数$dfs(x,y,nowcoin,mflag)$

为了方便判断，搜索前规定一下颜色，0代表无色 1代表红色 2代表黄色

搜索时无非两种情况：

1、下一个格子无色，如果魔法未使用，那就使用魔法过去

2、下一个格子有色，如果颜色相同，直接过去；否则，就花钱过去

dfs函数：

```
void mofa(int x, int y, int nowcoin, bool mflag)  //当前坐标x,y；当前使用的金币数nowcoin；是否使用魔法，0代表没有，1代表有
```

G.小木棍

根据课上PPT分析的几种剪枝，选择3种左右效率高一点的剪枝即可。

先自己做，代码略。

H.生日蛋糕

做这道题，你首先要有点数学常识：

首先圆柱的各个公式：
$V = \pi R^2 H$ 体积
$S_{侧} = 2\pi RH$ 侧面积
$S_{底} = \pi R^2$ 底面积

表面积=侧面积+底面积

这道题的难点主要在于dfs函数的设计，即搜索状态的寻找。

根据题目意思，涉及到体积、表面积、高度等搜索信息，为了准确转移搜索的信息和剪枝，这道题需要涉及五个参数

$dfs(sum, v, dep, h, r)$ ，sum表面涂的面积，v是蛋糕体积，dep是第几层蛋糕，h是目前蛋糕的高度，r是目前蛋糕的半径

除了dep是搜索的层次以外，其余的参数引入是为了辅助搜索和剪枝。

为了尽可能不搜索无效的表面积、体积、高度，主要有以下几种剪枝

- **搜索顺序上的剪枝**：从**体积大的搜到体积小，每次的半径至少减一。**（这样的话若是方案不合法在一个大的的时候就可以直接扔掉不要，但是如果是小的的话，可能就要试很多个。）
- **可行性剪枝1**：当 **当前的体积+之后预测到的自己认为的最优的体积>n** 的时候就可以直接舍弃这一个不合法的方案。（当前的加上最优的都已经是不合法的了，还能怎么办？）
- **可行性剪枝2**：上下界剪枝。根据多年的数学学习，**把半径和高度的冗余的状态全部都丢掉不要。**
- **最优性剪枝1**：在dfs的过程中可能会有很多次搜索到不合法的方案，若是不合法的方案用到的面积刚好等于n的时候（也就是说给定的面积全部都用完了），那么最优解的ans一定小于此时的答案，这个时候在放弃掉这个方案之前，可以对答案进行进一步的更新；
- **最优性剪枝2:** 若 **当前的面积+之后预测到的自己认为的最优的面积>**ans 的时候就可以直接舍弃这一个不优秀的方案，因为他没有目前的答案小。

由于以上的剪枝涉及到大量表面积、体积的计算，为了简化代码避免重复计算，这一部分可以先预处理两个数组$minv[]$和$minb[]$，节约搜索时间。

先自己做，代码略