



# 算法基础

郭炜 微博 <http://weibo.com/guoweiofpku>

<http://blog.sina.com.cn/u/3266490431>

刘家瑛 微博 <http://weibo.com/pkuliujiaying>



北京大学  
PEKING UNIVERSITY

信息科学技术学院《算法基础》 郭炜 刘家瑛

# 深度优先搜索

生日蛋糕

# 生日蛋糕 (POJ1190)

要制作一个体积为 $N\pi$ 的 $M$ 层生日蛋糕，每层都是一个圆柱体。

设从下往上数第 $i$  ( $1 \leq i \leq M$ )层蛋糕是半径为 $R_i$ ，高度为 $H_i$ 的圆柱。当 $i < M$ 时，要求 $R_i > R_{i+1}$ 且 $H_i > H_{i+1}$ 。

由于要在蛋糕上抹奶油，为尽可能节约经费，我们希望蛋糕外表面（最下一层的下底面除外）的面积 $Q$ 最小。

$$Q = S\pi$$

请编程对给出的 $N$ 和 $M$ ，找出蛋糕的制作方案（适当的 $R_i$ 和 $H_i$ 的值），使 $S$ 最小。  
（除 $Q$ 外，以上所有数据皆为正整数）

# 解题思路

- 深度优先搜索，枚举每一层可能的高度和半径。

# 解题思路

- 深度优先搜索，枚举每一层可能的高度和半径。
- 确定搜索范围：底层蛋糕的最大可能半径和最大可能高度

# 解题思路

- 深度优先搜索，枚举每一层可能的高度和半径。
- 确定搜索范围：底层蛋糕的最大可能半径和最大可能高度
- 搜索顺序：
  - 从底层往上搭蛋糕，而不是从顶层往下搭
  - 在同一层进行尝试的时候，半径和高度都是从大到小试

# 剪枝

- 剪枝1：搭建过程中发现面积超过已经求得的最优表面积，则停止搭建

# 剪枝

- 剪枝1：搭建过程中发现面积超过已经求得的最优表面积，则停止搭建
- 剪枝2：搭建过程中预见到再往上搭，高度已经无法安排，或者半径已经无法安排，则停止搭建



# 剪枝

- 剪枝1：搭建过程中发现面积超过已经求得的最优表面积，则停止搭建
- 剪枝2：搭建过程中预见到再往上搭，高度已经无法安排，或者半径已经无法安排，则停止搭建
- 剪枝3：搭建过程中发现还没搭的那些层的体积，一定会超过还缺的体积，则停止搭建

# 剪枝

- 剪枝1：搭建过程中发现面积超过已经求得的最优表面积，则停止搭建
- 剪枝2：搭建过程中预见到再往上搭，高度已经无法安排，或者半径已经无法安排，则停止搭建
- 剪枝3：搭建过程中发现还没搭的那些层的体积，一定会超过还缺的体积，则停止搭建
- 剪枝4：搭建过程中发现还没搭的那些层的体积，最大也到不了还缺的体积，则停止搭建

```

#include <iostream>
#include <vector>
#include <cstring>
#include <cmath>
using namespace std;

int N,M;

int minArea = 1 << 30; //最优表面积
int area = 0; //正在搭建中的蛋糕的表面积
int minV[30]; // minV[n]表示n层蛋糕最少的体积
int minA[30]; // minA[n]表示n层蛋糕的最少侧表面积
int main()
{
    cin >> N >> M ; //M层蛋糕，体积N
    minV[0] = 0;
    minA[0] = 0;
    for( int i = 1; i<= M; ++ i) {
        minV[i] = minV[i-1] + i * i * i; //第i层半径至少i,高度至少i
        minA[i] = minA[i-1] + 2 * i * i;
    }
    if( minV[M] > N )
        cout << 0 << endl;
}

```

```

else {
    int maxH = (N - minV[M-1]) / (M*M) + 1; //底层最大高度
    //最底层体积不超过 (N-minV[M-1]), 且半径至少M
    int maxR = sqrt(double(N-minV[M-1]) / M) + 1; //底层高度至少M
    area = 0;
    minArea = 1 << 30;
    Dfs( N, M, maxR, maxH );
    if( minArea == 1 << 30)
        cout << 0 << endl;
    else
        cout << minArea << endl;
}
}

```

```
void Dfs(int v, int n,int r,int h)
//要用n层去凑体积v,最底层半径不能超过r,高度不能超过h
//求出最小表面积放入 minArea
{
    if( n == 0 ) {
        if( v ) return;
        else {
            minArea = min(minArea,area);
            return;
        }
    }
    if( v <= 0)
        return ;
    if( minV[n] > v ) //剪枝3
        return ;
    if( area + minA[n] >= minArea) //剪枝1
        return ;
    if( h < n || r < n ) //剪枝2
        return ;
```

```

if( MaxVforNRH(n,r,h) < v )    //剪枝4
//这个剪枝最强！没有的话，5秒都超时，有的话，10ms过！
    return;
//for( int rr = n; rr <= r; ++ rr ) { 这种写法比从大到小慢5倍
for( int rr = r; rr >=n; -- rr ) {
    //从大到小快！这边大了，内部的循环可选择范围就少了
    //符合先做选择步骤少的思路
    if( n == M ) //底面积
        area = rr * rr;
    for( int hh = h; hh >= n ; --hh ) {
        area += 2 * rr * hh;
        Dfs(v-rr*rr*hh,n-1,rr-1,hh-1);
        area -= 2 * rr * hh;
    }
}
}

```

```
int MaxVforNRH(int n,int r,int h)
{ //求在n层蛋糕，底层最大半径r，最高高度h的情况下，能凑出来的最大体积
    int v = 0;
    for( int i = 0; i < n ; ++ i )
        v += (r - i ) *(r-i) * (h-i);
    return v;
}
```