1.5 回溯算法

主讲老师：党东

回溯算法的简单优化

在搜索算法中优化中，剪枝，就是通过某种判断，避免一些不必要的遍历过程，形象的说，就是剪去了搜索树中的某些“枝条”，故称剪枝。应用剪枝优化的核心问题是设计剪枝判断方法，即确定哪些枝条应当舍弃，哪些枝条应当保留的方法。

回溯算法的简单优化

剪枝算法按照其判断思路可分成两类:**可行性剪枝**及**最优性剪枝**

可行性剪枝

该方法判断继续搜索能否得出答案，如果不能直接回溯。

最优性剪枝

最优性剪枝，又称为上下界剪枝，是一种重要的搜索剪枝策略。它记录当前得到的最优值，如果当前结点已经无法产生比当前最优解更优的解时，可以提前回溯。

【例1】数的划分 --1284 可行性剪枝

【问题描述】

将整数n分成k份，且每份不能为空，任意两份不能相同(不考虑顺序)。

例如：n=7，k=3，下面三种分法被认为是相同的。

1，1，5; 1，5，1; 5，1，1;

问有多少种不同的分法。

【输入文件】

n，k (6 < n <= 200，2 <= k <= 6)。

【输出文件】

一个整数，即不同的分法。

【输入文件】

7 3

【输出文件】

4

【例1】数的划分 --1284 可行性剪枝

【核心代码】

```
void DFS(int ki,int minx,int snum)
        //ki指划分的份数  minx是当前份的最小值  snum是指剩余未划分的数量
{       if(ki==k-1){
                count++;
                return ;
        }
        for(int i=minx;i<=snum;i++)
        {
                if(i>snum/(k-ki) break;
                //剪枝（强大的功能）：可行性剪枝
        //1.每份的数量必须由小到大，排除1 5 2和 1 2 5一类重复
        //2. 提前判断，第ki份被划分后，剩余量>=未划分的(k-ki)份*(最低划分量)minx
                DFS(ki+1,i,snum-i); //搜索下一层
        }
}
```

【例2】生日蛋糕--1488 可行性剪枝+最优化剪枝

【问题描述】

7月17日是Mr.W的生日，ACM-THU为此要制作一个体积为Nπ的M层生日蛋糕，每层都是一个圆柱体。

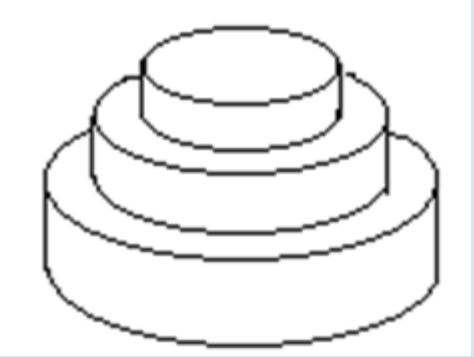
设从下往上数第i(1<=i<=M)层蛋糕是半径为R[i], 高度为Hi的圆柱。当i < M时，要求 R[i]>R[i+1]且H[i]>H[i+1]。

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。

令Q= Sπ

请编程对给出的N和M，找出蛋糕的制作方案（适当的Ri和Hi的值），使S最小。

（除Q外，以上所有数据皆为正整数）



【输入文件】

有两行，第一行为N（N<=10000），表示待制作的蛋糕的体积为Nπ；第二行为M(M<=20)，表示蛋糕的层数为M。。

【输出文件】

仅一行，是一个正整数S（若无解则S=0）。

【例2】生日蛋糕--1488 可行性剪枝+最优化剪枝

【输入样例】
100
2
【输出样例】
68
【样例说明】
附：圆柱公式
体积$V=\pi R^2H$
侧面积$A'=2\pi RH$
底面积$A=\pi R^2$

【例2】生日蛋糕--1488 可行性剪枝+最优化剪枝

【题目大意】
制作一个体积为Nπ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<=i<=M）层蛋糕是半径为Ri，高度为Hi的圆柱体。当i<M时，要求Ri>Ri+1且Hi>Hi+1。
希望蛋糕外表面（最下层的下底面除外）的面积最小

【例2】生日蛋糕--1488 可行性剪枝+最优化剪枝

【常规剪枝】
mins 表示i层最小侧面积
minv 表示i层最小体积
显然易见的剪枝：
$S_{i-1}+2R_iH_i+mins[m-i]>=ans;$
$D_{i-1}+R_i2H_i+minv[m-v]>N;$

【例2】生日蛋糕--1488 可行性剪枝+最优化剪枝

【另外一个剪枝】

■v表示已占用的体积

■s表示已用奶油面积

2 * (n - v) / r + s >= ans 这是该题的精髓，如果没有的话会造成超时，是为了把v和s联系起来，原因如下:

假设能够得到最小值时(为什么这样假设呢，因为如果得不到的话那么就已经被第一个剪枝滤去了,所以在第三个剪枝验证时表示已经通过了第一个剪枝的要求)，

n - v= h[1] * r[1] * r[1] + … + h[p] * r[p] * r[p] < h[1] * r[1] * r + … + h[p] * r[p] * r （因为r是p + 1层的半径)

其中h[1]...h[p]表示在函数的形参情况下，1到p层应该取得h值，r[1]同理

两边同时处以r 再乘以2得 2 * (n - v) / r < 2 * (h[1] * r[1] + … + h[p] * r[p])

2 * (n - v) / r < ans - s

2 * (n - v) / r + s < ans 成立，则可得剪枝条件

【例2】生日蛋糕--1488 可行性剪枝+最优化剪枝

【核心代码】

```
const int INF=2147483647;
int mins[21],minv[21],m,n,ans;
void search(int v,int s,int p,int r,int h)//v-0,s-0,p-m,r-n+1,h-n+1
{
    int i,j,hh;
    if(p==0)//蛋糕已完成
      {
            if(v==n&&s<ans)//判断是否符合要求并得到更优解
                  ans=s;//更新最优解
            return ;
      }
    if(v+minv[p-1]>n)//判断此方案是否可行
            return ;
    if(s+mins[p-1]>ans)//判断此方案是否可行
            return ;
    if(2*(n-v)/r+s>=ans)//判断此方案是否优于最优解
            return ;
    for(i=r-1;i>=p;i--)
      {
            if(p==m)//如果是最底层，则面积要加上上底面
            s=i*i;
            hh=min((n-v-minv[p-1])/(i*i),h-1);//通过数学公式计算更优h,最优化剪枝
            for(j=hh;j>=p;j--)//搜索上一层的高
            search(v+i*i*j,s+2*i*j,p-1,i,j);//递归求上一层的高
      }
}
```

【例2】生日蛋糕--1488 可行性剪枝+最优化剪枝

【核心代码】

```
int main()
{
    cin>>n>>m;
    ans=INF;
    mins[0]=minv[0]=0;
    for(int i=1;i<21;i++)
     {          //此题所有半径和高度都是正整数，所以可得下面的式子
                mins[i]=mins[i-1]+2*i*i;//mins表示从最上面一层到 i 层的最小表面积(这里仅仅算了侧面)
                minv[i]=minv[i-1]+i*i*i; //minv表示从最上面一层到 i 层的最小体积
     }
    search(0,0,m,n+1,n+1);
    if (ans==INF) cout<<"0";
    else cout<<ans;
    return 0;
}
```