算法分析与设计第二次实验报告

问题描述

设有一个长度为L的钢条，在钢条上标有n个位置点（p1,p2,...,pn）。现在需要按钢条上标注的位置将钢条切割为n+1段，假定每次切割所需要的代价与所切割的钢条长度成正比。请编写一个算法，能够确定一个切割方案，使切割的总代价最小。

问题分析

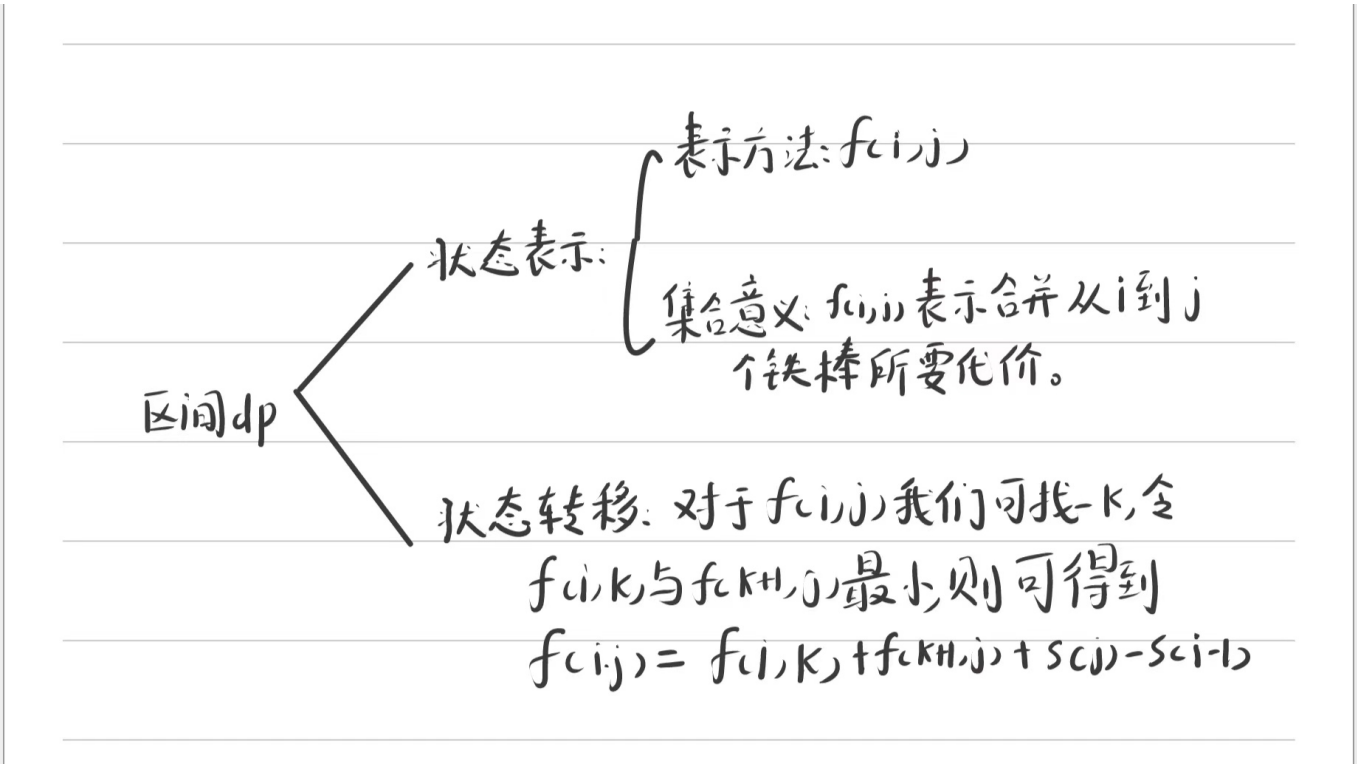
这是一个区间dp问题，具体可以归类到石子划分问题。题目的要求是将一根铁棒不断在给定的位置切割成n+1段，并消耗一定的代价，我们不妨反向思考问题--现在有n+1段铁棒，我们要将这些铁棒合成一个大铁棒，并且对于任意两个铁棒A和B，如果我们想将他们合并，需要付出的代价为：

$$cost = length(A) + length(B)$$

而且显然，我们只能合成相邻的两段铁棒。例如对于三根按照切割位置摆放的铁棒A,B,C，为了使得最后的合并满足题目中对于切割位置的要求，不可能合成A和C，而只能合成B和C，或者A和B。

由此我们可以将问题简化成一个区间dp问题。

下面是思路简写。



其中对于查找$f(i,j)$,我们可以找到一个分界点k,那么很显然
$f(i,j)=f(i,k)+f(k+1,j)+s(j)-s(i-1)$,
只要$f(i,k)$和$f(k+1,j)$都取到了最小，那么$f(i,j)$当然也是最小。

算法设计

首先划定好前缀和数组。
其次我们先枚举划分的区间长度(len)，对于所有合法的len我们都用进行一次计算。
接着利用双指针，第一个指针指向第一个小铁棒。第二个指针是末尾指针，位置是当前len下，位置在该区间最后一个的铁棒的下标，即$i+len-1$。
对于两个指针之间的范围，我们不断枚举其中的所有k，找到令$f(i,j)$最小的位置，并更新$f(i,j)$的值。

由我们对$f(i,j)$的定义可知，答案就是$f(1,n+1)$。

我们一共要枚举$(n+1)$个区间，而每个区间又要进行至少$（n+1）^2$次操作，因此复杂度为$O(n^3)$。

代码实现

```
#include <iostream>

#include <string>


#include <cmath>

#include <vector>

#include <algorithm>

using namespace std;

void MinCost(int L,int n,int *p)//n+1份，节点位置存在p里面
{

        int s[10000]={0};
        sort(p,p+n+1);
        int f[300][300];
        for(int i=0;i<211;i++)
        for(int j=0;j<210;j++)
```

```
        f[i][j]=0;

        for(int i=1;i<=n+1;i++)
        {
                s[i]=p[i]-p[i-1];
                s[i]+=s[i-1];
        }

        for(int len=2;len<=n+1;len++)
        {
                for(int i=1;i+len-1<=n+1;i++)
                {
                        int j=i+len-1;
                        f[i][j]=1e8;
                        for(int k=i;k<j;k++)
                        {
                                f[i][j]=min(f[i][j],f[i][k]+f[k+1][j]+s[j]-s[i-1]);

                        }
                }
        }
        cout<<f[1][n+1]<<endl;

}

//你的代码只需要补全上方函数来实现算法,可根据自己需要建立别的函数

//其中L是钢条长度，n是位置点个数，p包含n 个切割点的位置（乱序）

//只需要提交这几行代码，其他的都是后台系统自动完成的。类似于 LeetCode
```

运行结果

平台提供用例：

试答	状态	成绩 / 10.00	回顾
1	已结束 已提交 2023年11月5日 星期日 21:57	9.70	回顾
2	已结束 已提交 2023年11月5日 星期日 21:58	10.00	回顾

其他oj提供用例测评：

作者： csranking2022　　结果： Accepted

通过了 11/11个数据　　运行时间： 95 ms

运行空间： 596 KB　　提交时间： 6秒前

延伸拓展

不难发现本题背景为石子合并问题，隶属于区间合并的一种。

设有 N 堆石子排成一排，其编号为 $1, 2, 3, \ldots, N$。

每堆石子有一定的质量，可以用一个整数来描述，现在要将这 N 堆石子合并成为一堆。

每次只能合并相邻的两堆，合并的代价为这两堆石子的质量之和，合并后与这两堆石子相邻的石子将和新堆相邻，合并时由于选择的顺序不同，合并的总代价也不相同。

例如有 4 堆石子分别为 `1 3 5 2`， 我们可以先合并 1、2 堆，代价为 4，得到 `4 5 2`， 又合并 1、2 堆，代价为 9，得到 `9 2` ，再合并得到 11，总代价为 $4+9+11=24$；

如果第二步是先合并 2、3 堆，则代价为 7，得到 `4 7`，最后一次合并代价为 11，总代价为 $4+7+11=22$。

问题是：找出一种合理的方法，使总的代价最小，输出最小代价。

给出一段代码如下，思路与本次试验相同。

```
# include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=310;
int f[N][N];
int s[N];
int n;
int main (void)
{
        cin>>n;
        for(int i=1;i<=n;i++)
        {
                cin>>s[i];s[i]+=s[i-1];
```

```
        }

        for(int len=2;len<=n;len++)
        {
                for(int i=1;i+len-1<=n;i++)
                {
                        int j=i+len-1;
                        f[i][j]=1e8;
                        for(int k=i;k<j;k++)
                        {
                                f[i][j]=min(f[i][j],f[i][k]+f[k+1][j]+s[j]-s[i-1]);

                        }

                }
        }
        cout<<f[1][n]<<endl;
        return 0;

}
```