动态规划 - 实验2报告

李雨轩
2204112913
计算机2205

表1

一、题目描述

有一根长度为L的钢条，在钢条上有n个位置点（$p_1, p_2, \cdots, p_n$）。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。求最小的切割总代价。

二、问题分析

拿到问题后，第一反应是看能否使用贪心算法快速求解。但我们发现，每次选择切点的时候，会对其两侧的点切割时的代价产生影响，并没有合适的贪心策略。故我们只能放弃贪心算法，寻找其他算法。 仔细分析题目，我们发现：当我们计算 $[p_l, p_r]$ 区间内的切割的总代价时，我们可以从中选择任意一个切点将其切开，然后分别计算切点左侧和右侧的最小总代价，将他们相加，再加上从p_l到p_r的总长度，即本次切割的代价，就可以得到在从这个点切割后，$[p_l, p_r]$ 区间的总代价。我们再便利区间中所有的切割点的总代价，找到其中最小的一个就是要求的最小总代价。 我们发现，这个题要求的是当总体取到最优值时，其递推公式的各个部分也取到最优值,满足最优子结构性质。因此，我们可以使用动态规划算法来解决问题。

三、算法设计

确定了要用动态规划算法之后，如何确定状态和推导出状态转移方程就成为了解决问题的重中之重。显然，我们可以将状态设为 区间的最小切割总代价，根据上面的推导，我们不难得到动态转移方程如下：

$$\mathrm{dp}[i][j] = \begin{cases} 0 & \text{if } i = j \text{ or } i = j+1 \\ \min_{i<k<j} \left(\mathrm{dp}[i][k] + \mathrm{dp}[k][j] + (p[j] - p[i])\right) & \text{otherwise} \end{cases}$$

其中：

- $\mathrm{dp}[i][j]$ 表示将位置点 i 到位置点 j 之间的钢条切割成最小段的最小代价。
- $p[]$ 存储了位置点的位置，已经排好序。

- $p[j]-p[i]$ 表示从位置点 i 到位置点 j 之间的钢条长度。
- 初始状态为 $\mathrm{dp}[i][i]=\mathrm{dp}[i][i+1]=0$，表示单个点和相邻两个点之间的切割都不需要代价。
- 通过枚举位置点 k，计算出在 $[i,j]$ 区间内的所有切割方式的代价，选择代价最小的切割方式作为 $\mathrm{dp}[i][j]$ 的值。

这个状态转移方程描述了如何从小的问题构建出大的问题的解，以达到确定切割钢条的最小代价的目的。下面我们来计算该算法的时间复杂度：

对于每个$dp[i][j]$我都需要循环$j-i$次来寻找能取到最小总价值的k。因为i取值范围为$1-n$，j取值范围为$i-n$，所以不难发现：

$$N=\sum_{i=1}^{n}\sum_{j=i}^{n}j-i$$

经过计算不难得到:

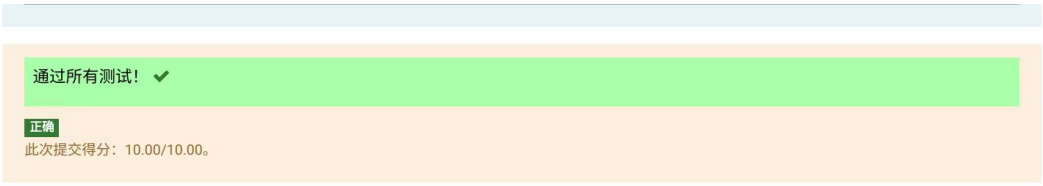
$$O(n)=n^3$$

四、算法实现：源代码

```
...

void MinCost(int L, int n, int *p)
{
  sort(p + 1, p + n + 1);
  auto dp = new int *[n + 2];
  for (int i = 0; i < n + 2; i++)
  {
    dp[i] = new int[n + 2];
    for (int j = 0; j < n + 2; j++)
      dp[i][j] = 0x7fffffff;
  }

  for (int i = 0; i < n + 1; i++)
  {
    dp[i][i] =  dp[i][i + 1] = 0;
  }
  dp[n + 1][n + 1] = 0;

  for (int k = 1; k < n + 2; k++)
  {

    for (int i = 0; i + k < n + 2; i++)
```

```
    {
      int l = p[i + k] - p[i];
      for (int j = i + 1; j < i + k; j++)
        dp[i][i + k] = min((dp[i][j] + dp[j][i + k] + l), dp[i][i
+ k]);
    }
  }
  cout << dp[0][n + 1] << endl;
  return;
}

...
```

五、运行结果：Moodle平台测试用例结果




```
rouge@debian ~/XJTUALGO/Labs  main g++ MinCost.cpp -o MinCost && ./MinCost
7 4
1 3 4 5
16
rouge@debian ~/XJTUALGO/Labs  main
```

六、实验过程说明

在我们进行这次实验时，我们的首要目标是解决钢条切割问题，并找出最小代价的解决方案。一开始，我们采用了贪心算法来解决这个问题，但很快发现并没有找到一个合适的贪心策略。因此，我们转而考虑了动态规划算法。我们开始对问题进行深入分析，确认了动态规划算法的适用性。接下来，我们着手设计了算法，将问题转化为状态及状态转移方程的形式。通过这一过程，我们确定了动态规划算法的关键实现步骤。然后，我们着手编写算法的代码，并进行了调试和测试工作。在测试阶段，我们运用了Moodle平台提供的测试用例对算法的各种情况进行了验证，得到了相应的运行结果。