**深 圳 大 学 课 程 设 计 报 告**

**课程名称：­ 数据结构**

**项目名称： 石子合并**

**任课教师： 杨艳丽**

**学 院： 计算机与软件学院**

**专 业： 计算机科学与技术**

**组员1 姓名： 刘清影 学号： 2017152021**

**组员2 姓名： 学号：**

**组员3 姓名： 学号：**

|  |
| --- |
| 教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |

目录

[问题描述 1](#_Toc532075737)

[算法描述 1](#_Toc532075738)

[动态规划的正确性 2](#_Toc532075739)

[动态规划的描述 3](#_Toc532075740)

[时间复杂度 3](#_Toc532075741)

[平行四边形优化 3](#_Toc532075742)

[其他优化 4](#_Toc532075743)

[算法实现 4](#_Toc532075744)

[实验结果 5](#_Toc532075745)

[问题分析 7](#_Toc532075746)

[心得与体会 7](#_Toc532075747)

[任务分工及所完成的状况 7](#_Toc532075748)

[参考文献 7](#_Toc532075749)

[源代码 7](#_Toc532075750)

石子合并

刘清影

2018/12/8 20:00:00

## 问题描述

在一个圆形操场的四周摆放着n 堆石子。现要将石子有次序地合并成一堆。规定每次只能选相邻的2 堆石子合并成新的一堆，并将新的一堆石子数记为该次合并的得分。求将n堆石子合并成一堆的最小得分和最大得分。

要求：

- 采用的算法：动态规划

- 要求给出计算表格和结果

## 算法描述

读完题目之后，我们可以发现题目具有这两个特点：

1. 每次只能合并相邻两堆石子
2. 圆形操场

特点1限制了此题不可以使用贪心算法，而应该使用更加准确的算法——动态规划，具体地说，是区间类型的动态规划。而对于特点二，我们采用“断链为环”的思想去解决这个问题。

### 动态规划的正确性

前面已经说过，虽然这一道题目的描述让人非常容易联想到贪心算法，但是应用贪心算法来解决这个问题是错误的，可以看下面这个例子：

**Example**

我们假如5堆的石子，其中石子数分别为7，6，5，7，100，我们假设求合并的最小得分。

- 按照贪心算法，合并的过程如下：

1. 第一次合并7，6，5，7，100，结果为5+6=11
2. 第二次合并7，11，7，100，结果为7+11=18
3. 第三次合并18,7,100, 结果为18+7=25
4. 第四次合并25,100，结果为100+25=125

此时，贪心思想的合并已经结束，总得分为：11+18+25+125=179

- 另外一种方案：

1. 第一次合并7，6，5，7，100，结果为7+6=13
2. 第二次合并13，5，7，100，结果为5+7=12
3. 第三次合并13,12,100, 结果为13+12=25
4. 第四次合并25,100，结果为100+25=125

此时，另外方案的合并已经结束，总得分为：13+12+25+125=175

通过对比可以发现，显然此题使用贪心算法是不正确的。

对于本题，很显然满足动态规划的最优子结构和和重叠子问题的性质，所以我们可以考虑使用动态规划求解。

最优子结构的证明: 如果一个区间的最大得分是,那么它必然是由两个最大得分的子区间(,)的得分合并而成，即 否则，会存在一个更大的子区间的得分,使得 这和我们假设是区间的最大得分矛盾，所以不存在一个更大子区间的得分。所以最大得分的子区间的得分也是子区间的最大得分。

最小得分同上理。

至于重叠子问题，是比较显然的，因为区间的计算依赖于子区间的计算。也就是说，区间的计算可以把它看成两个子区间的石子的合并。

通过以上分析，动态规划是适合解决这道问题的。

### 动态规划的描述

其实在前面就已经多次提到了把这一道题目“区间化”，也就是说，把两次合并的石子看出是由两个区间合并而成的结果。我们假设区间的最大得分为，最小得分为，那么有：

dpmax(i,j) = max(dpmax(i,k)+dpmax(k+1,j)+sum(i,j)), {1}

dpmin(i,j) = min(dpmin(i,k)+dpmin(k+1,j)+sum(i,j)) , {2}

上面的公式表达了把区的计算转换成了子区间的计算，所以我们可以通过迭代的方式或者记忆化搜索的方式进行计算。

### 时间复杂度

因为根据表达式可以知道，存在三个遍历，我们需要用三重循环去遍历，故时间复杂度为。如果n的范围是，那么这需要跑20多分钟才能运行完毕（100组数据），显然这样太慢了。幸运的是，我们可以使用平行四边形优化去把的时间复杂度降低到，让20分钟的计算时间压缩到5秒钟就可以完成。

### 平行四边形优化

在**Efficient Dynamic Programming Using Quadrangle Inequalities**这篇文章中提到了如何将

c(i,j) = w(i,j) + min(c(i,k)+c(k+1,j)) {3}

的类型的动态规划的时间复杂度降低到。我在这里做一个梳理，具体证明请到原文查阅。

**定理1**： 如果式的满足平行四边形不等式,即

w(i,j)+w(i',j')<=w(i,j')+w(i',j) i<=i'<=j<=j' {4}

且满足区间单调性

w(i,j)<=w(i',j') ，[i，j] [i',j'] {5}

那么的动态规划的计算可以加速到。

**定理2**： 如果式的满足定理(1)，那么也满足平行四边形不等式.

**定理3**： 如果式的满足平行四边形不等式，那么有

K\_c(i,j)<=K\_c(i,j+1)<=K\_c(i+1,j+1) {6}

其中表示最大的使得最小值成立。

显然，我们计算最小得分式满足平行四边形不等式，所以我们可以把k优化成式，使得计算最小得分的动态规划的时间复杂度为

### 其他优化

对于最大值，根据https://www.luogu.org/blog/Hurricane-zjz/solution-p1880 这篇文章可以知道，最大值存在这样的性质：即总是在两个端点的最大者中取到。即

dpmax(i,j)=max(dpmax(i,j),dpmax(i+1,j))+sum(i,j) {7}

还有就是在计算的时候使用前缀和处理，这样就不用每次都用循环计算的和了。

## 算法实现

使用二维数组和来保存子区间的计算结果，二维数组保存的值，保存前缀和。通过遍历区间的起始位置和长度,根据式(1)和(2)来计算即可。值得注意的是，区间的长度应该式从小到大，因为我们需要先计算子问题再去计算大问题。

断链为环： 将储存石子分数的数组延长两倍，使其每个石子都可以访问相邻两边的石子，最后通过动态规划计算完成之后，我们通过遍历起始位置在[1,n]，区间长度为n的dpmax或者dpmin就可以得到最大分数或者最小分数。比如在下面的表格（石子个数为4）中，红色部分表示我们遍历的元素：

0 | 8 | 21 | 43 | 60 | 76 | INF | INF |  
  
 0 | 0 | 9 | 27 | 44 | 60 | INF | INF |  
  
 0 | 0 | 0 | 14 | 31 | 44 | INF | INF |  
  
 0 | 0 | 0 | 0 | 13 | 25 | INF | INF |  
  
 0 | 0 | 0 | 0 | 0 | 8 | INF | INF |  
  
 0 | 0 | 0 | 0 | 0 | 0 | INF | INF |  
  
 0 | 0 | 0 | 0 | 0 | 0 | 0 | INF |  
  
 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 实验结果

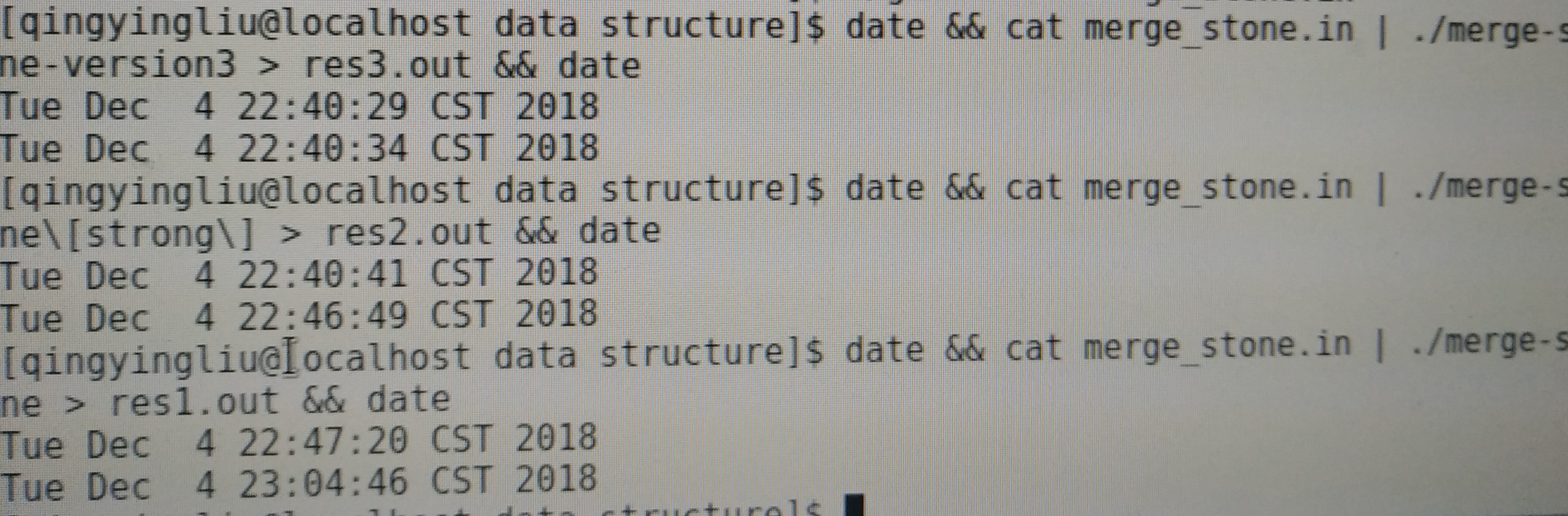
假设我们输入4个石子，它们的得分相应是4,4,5,9，那么最大值是54，最小值是43. 的表格是：

0 | 8 | 21 | 43 | 60 | 76 | INF | INF |  
  
 0 | 0 | 9 | 27 | 44 | 60 | INF | INF |  
  
 0 | 0 | 0 | 14 | 31 | 44 | INF | INF |  
  
 0 | 0 | 0 | 0 | 13 | 25 | INF | INF |  
  
 0 | 0 | 0 | 0 | 0 | 8 | INF | INF |  
  
 0 | 0 | 0 | 0 | 0 | 0 | INF | INF |  
  
 0 | 0 | 0 | 0 | 0 | 0 | 0 | INF |  
  
 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

的表格是：

0 | 8 | 22 | 54 | 80 | 110 | 0 | 0 |  
  
 0 | 0 | 9 | 32 | 54 | 80 | 0 | 0 |  
  
 0 | 0 | 0 | 14 | 32 | 54 | 0 | 0 |  
  
 0 | 0 | 0 | 0 | 13 | 30 | 0 | 0 |  
  
 0 | 0 | 0 | 0 | 0 | 8 | 0 | 0 |  
  
 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  
  
 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  
  
 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

在运行时间方面，对于100组数据n在1000之内的数据，没有优化过的动态规划运行了17分钟，只用平行四边形优化的跑了6分钟，用平行四边形优化和最大的的性质优化的跑了5秒钟。如下图所示：



第一组是：用平行四边形优化和最大的的性质优化

第二组是：只用平行四边形优化的

第三组是：没有优化过的动态规划

## 问题分析

1. 在使用动态规划的时候应该先计算子问题，在我们这一题，我们应该根据区间的长度由小到大地去计算.
2. 在使用平行四边形优化的时候需要注意边界条件，特别是注意等号，如果没有等号，会出现错误的结果。

## 心得与体会

通过本次实验，我学会了使用动态规划去解决问题，以及分析它的时间复杂度。最重要的是，我学会了如何去把平行四边形优化应用到动态规划。 在本次实验中，我去解释和证明了石子合并的最优子问题和重叠子问题，然后说明了动态规划的解决方法，最后给出了三个定理去把动态规划优化到。在写这次作业之前，我还特意去看了平行四边形优化的论文，那里给出了严格的数学证明，让人看了收益匪浅。

## 任务分工及所完成的状况

本次任务只有我刘清影一人完成，题目任务要求全部完成，在此基础上我还加了平行四边形优化，让的时间复杂度优化到，优化后的效果非常好，让十几分钟的程序优化到几秒钟就能完成。

## 参考文献

[1] “Efficient Dynamic Programming Using Quadrangle Inequalities ”， F. Frances Yao Xerox Palo Alto Research Center Palo Alto, California ， http://www.cs.ust.hk/mjg\_lib/bibs/DPSu/DPSu.Files/p429-yao.pdf

## 源代码

源代码在我的github上：https://github.com/Lewin671/algorithm/tree/master/Data%20structure/project/code 其中version1是没有优化过的。version2是只有平行四边形优化的，version3是平行四边形+最大值的特性优化的

**个人得分权重分配表**

|  |  |  |  |
| --- | --- | --- | --- |
| 排序 | 姓名 | 学号 | 项目个人权重 |
| 1 |  |  | % |
| 2 |  |  | % |
| 3 |  |  | % |

我组成员总共\_\_\_\_\_\_\_名，权重总和为： %

**本组成员郑重承诺在课程设计完成过程中不发生任何不诚信现象，一切不诚信所导致的后果均由本组成员承担。同时我组成员同意此课程设计的个人得分权重分配表。**

**签名（手签全部成员）：**