第四章作业 2203202 2022111189 李博文.

第四章作业

4. 给定一个整数序列$a_1,\ldots,a_n$。相邻两个整数可以合并，合并两个整数的代价是这两个整数之和。通过不断合并最终可以将整个序列合并成一个整数，整个过程的总代价是每次合并操作代价之和。试设计一个动态规划算法给出$a_1,\ldots,a_n$的一个合并方案使得该方案的总代价最大。设计动态规划算法求解此问题并分析算法的时间复杂性。

先定义一个二维数组 dp[i][j] 表示：从 i 到 j 的最大合并代价.
先初始化 dp，使所有位置元素都为0，然后读取整数序列.
对于每个子序列长度计算，求出最大合并代价.
每个子序列，递归分割，计算各子序列的合并代价之和，后更新 dp
最后，dp[0][n-1] 就是最大合并代价

代码：

```
def max_range_cost(array):
    n = len(array)
    dp = [[0]*n for _ in range(n)]
# 遍历
    for length in range(2, n+1):  # 子列长
        for i in range(n-length+1):
            j = i + length - 1
            for k in range(i, j):  # 所有可分割点
                cost = arr[k+1] + arr[j]
                if k == i:  # 边界
                    dp[i][j] = cost  # 最大
                else:  # 否则加上前一部分的代价
                    dp[i][j] = max(dp[i][j], dp[i][k] + cost)
    return dp[0][n-1]

array = [int(x) for i in input().split()]
print(max_merge_cost(array))
```

时间复杂度分析：一次递归和次遍历，时间复杂度为 $O(n^3)$