东南大学
网络空间安全学院

算法作业 #2

学生姓名: 李盛; 学号: *229221*

Course: 算法设计与分析 – Professor: 陶军
Due date: *4* 月 *8* 日, *2022* 年

第一题

1. 给出 N 个 1-9 的数字 $(v_1, v_2, \ldots, v_N)$, 不改变它们的相对位置, 在中间加入 K 个乘号和 $N-K-1$ 个加号, (括号随便加) 使最终结果尽量大。因为乘号和加号一共就是 $N-1$ 个了所以恰好每两个相邻数字之间都有一个符号。并说明其具有优化子结构性质及子问题重叠性质。

Solution.

(*a*) **Optimal substructure**
For our first step in the dynamic-programming paradigm, we find the optimal substructure and then use it to construct an optimal solution to the problem from optimal solutions to subproblems.
Given N numbers (ranging from 1 to 9), we define the $N-1$ positions between two joint numbers as $p_1, p_2, \ldots, p_{N-1}$. If an optimal solution inserts K multiplication at positions p_{i_1}, $p_{i_2}, \ldots, p_{i_K}$ where $1 \le i_j \le N-1$ for $j = 1, 2, \ldots K$, then maximum corresponding result is:

$$r[N,K] = r[i_K, K-1] \times \sum_{i=i_k+1}^{N} v_i$$

We thus obtain the following simpler version of transfer equation:

$$r[N,K] = \max_{K \le i_k \le N-1} \left(r[i_k, K-1] \times \sum_{i=i_k+1}^{N} v_i \right)$$

Proof: solutions to the subproblems are optimal
Assume $r(i_k, K-1)$ is not optimal, then we have $r'(i_k, K-1) > r(i_k, K-1)$. Thus $r'(i_k, K-1) \times \sum_{i=i_k+1}^{N} v_i > r(i_k, K-1) \times \sum_{i=i_k+1}^{N} v_i$. We could conclude that $r'(n, K)$ can not be larger than $r(n, K)$ by contradiction.
Overlapping subproblems
$(v_1 + v_2 + v_3) \times v_4 \times (v_5 \times \ldots \times v_n)$
$(v_1 + v_2 + v_3) \times (v_4 + v_5) \times (v_6 \times \ldots \times v_n)$
For example, we have r[3, 0] in common and so on.

(*b*) Pseudocode

Algorithm 1 Bottom-Up-Mul-Add(A,N,K)

```
if K == 0 then
    return ∑_{i=1}^{N} A[i]
end if
Let r[1...N][0...K] be a new array
r[1][0] = 1
for n_i = 2...N do
    r[n_i][0] = r[n_i − 1][0] + n_i
end for
for k_i = 1...K do
    for n_i = k_i + 1...N do
        for n_j = k_i...n_i − 1 do
            r[n_i][k_i] = max(r[n_i][k_i], r[n_j][k_i − 1] × ∑_{i=n_j+1}^{n_i} A[i])
        end for
    end for
end for
```

(*c*) Implementation code with Python

```python
class Solution(object):
    def sum_A(self, A, l, r):
        res = 0
        for i in range(l, r+1):
            res += A[i]
        return res

    def Bottom_Up_Mul_Add(self, A, N, K):
        if K == 0:
            sumA = 0
            for i in A:
                sumA = sumA + i
            return sumA
        # init K = 0, i.e. without Multiplication
        r = [[0] * (K+1) for _ in range(N+1)]
        r[1][0] = 1
        for ni in range(2, N+1):
            r[ni][0] = r[ni-1][0] + ni

        for ki in range(1, K+1):
            for ni in range(ki+1, N+1):
                for nj in range(ki, ni):
                    r[ni][ki] = max(r[ni][ki], \
                    r[nj][ki-1]*self.sum_A(A,nj,ni-1))
        return r[N][K]
```

第二题

给定一长度为 N 的整数序列 $(a_1, a_2, \ldots, a_N)$，将其划分成多个子序列 (此问题中子序列是连续的一段整数)，满足每个子序列中整数的和不大于一个数 B，设计一种划分方法，最小化所有子序列中最大值的和。说明其具有最优化子结构及子问题重叠性质。
- 例如: 序列长度为 8 的整数序列 $(2,2,2,8,1,8,2,1), B = 17$，可将其划分成三个子序列 $(2,2,2)$，$(8,1,8)$ 以及 $(2,1)$，则可满足每个子序列中整数和不大于 17，所有子序列中最大值的和 12 为最终结果。

Solution.

(*a*) Definition: $A[i,j] = (a_i, a_{i+1}, \ldots a_j)$, $sum_{i,j} = a_i + a_{i+1} + \ldots + a_j$, where $1 \leq i \leq j \leq N$. Assume $A[i,j]$ is optimally divided into multiple subsequences such that the sum of the maximum of each subsequence $(r[i,j])$ is minimal. Then, the optimal solution to the length of N array $A[1,N]$ is:

$$r[1,N] = \min_{1 \leq k \leq N-1} (r[1,k] + max(A[k+1,N]))$$

where $sum_{k+1,N} \leq B\ \forall k, 1 \leq k \leq N-1$.

Optimal Structure

The optimal solution $r[1,N]$ involves the optimal solution to its subproblem r[1, k], thus it has the optimal structure.

Overlapping Subproblems

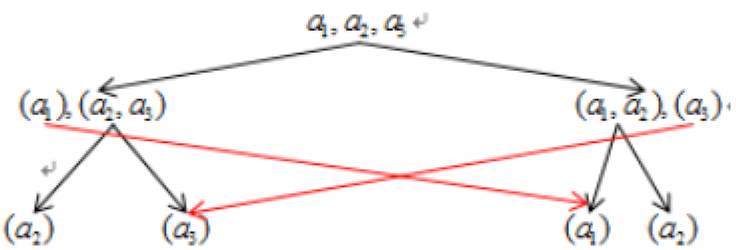

Figure 1: Example illusion of overlapping subproblems

As shown in Figure 1, we reuse r[1,1] and r[2,2] in the recursion tree to computer[1,3].

(*b*) Pseudocode

Algorithm 2 Min-maximums(A, N, B)

Let $r[1 \ldots N]$ be a new array.
$r[1] = A[1]$
for $i = 2 \ldots N$ **do**
  for $j = i \ldots 1$ **do**
    if $sum_{j,i} \leq B$ **then**
      $r[i] = min(r[i], r[j-1] + max(A[j,i]))$
    end if
  end for
end for

(c) Implementation code with Python

```python
class Solution(object):
    def sum_A(self, A, l, r):
        res = 0
        for i in range(l, r + 1):
            res += A[i]
        return res

    def min_maximums(self, A, N, B):
        if N == 1:
            return A[1]

        if N > 1:
            # A[0] = 0 for auxiliary
            A.insert(0,0)
            r = [2**40 for _ in range(N+1)]
            r[0] = 0
            r[1] = A[1]
            # compute r[2]...r[N]
            for i in range(2, N+1):
                j = i
                # move index j = i to the left
                while j >= 1 and self.sum_A(A, j, i) <= B:
                    r[i] = min(r[i], r[j-1] + max(A[j:i+1]))
                    j -= 1
            return r[N]
```