Algorithm Design Assignment 3

Jianan Lin, 662024667, linj21@rpi.edu

Problem 6.12. There are n servers. We can place a copy file in some servers and each server has a cost c_i. We will search this file from server 1 to n. If there is no such file in this server i, then it will turn to help from a server behind it until there is a server j with this file. The search cost is $j - i$. The last server must store this file. Design a polynomial efficient algorithm to get the minimum total cost.

Solution.

First we will give the algorithm and then explain and analyze it. Notice the index in an array with length $n + 1$ is from 0 to n.

Algorithm 1 DP1

```
OPT = [∞ × (n + 1)]
OPT[0] = 0
for i ∈ [1, n] do
    for j ∈ [0, i − 1] do
        OPT[i] = min( OPT[i],   c_i + OPT[j] + (1/2)(j − i + 1)(j − i) )
    end for
end for
return OPT[n];
```

1. Subproblem: Here, OPT_i means the least total cost from server 1 to i including i if we require that server i should have the copy. If $i = 0$, then obviously the cost is 0 because there is no server in this case. And if $i = n$, then OPT_i is what this question wants, i.e. the minimum total cost for these n servers.

2. Recurrence: For example, we assume that we have solved the cases from OPT_1 to OPT_{i-1}, i.e., for $\forall j < i$, we have found the minimum total cost from server 1 to j. Then we consider OPT_i. If the optimal case has only one server i with the file, then the total cost is $OPT_0 + (i - 1) + (i - 2) + ... + 1$. If the optimal case has at least 2 servers, suppose the last server with the file except i itself is k. Then the total cost is $OPT_k + (i - k - 1) + (i - k - 2) + ... + 1$. Therefore we prove the correctness and have the following formula:

$$OPT_i = \min_{0 \le j < i} \Big(c_i + OPT_j + (1 + ... + j - i) \Big) = \min_{0 \le j < i} \left(c_i + OPT_j + \frac{1}{2}(j - i)(j - i + 1) \right)$$

3. Complexity: Obviously the time complexity is $O(n^2)$ and the space complexity is $O(n)$.

Problem 6.27. There are n days and each day we need g_i gallon gas. The price to order 1 gallon gas is fixed and each time we order the gas, we should pay a fixed delivery fee P. It costs c to store 1 gallon gas for an extra day. We can store at most L gallons one time. Design an algorithm to decide when to order and how many gallons in each order to minimize the total cost.

Solution.

First we will give the algorithm and then explain and analyze it. Here "for $i \in [n - 1, 0]$" means we start from $i = n - 1$ and end with $i = 0$. Also notice that the index is from 0 to n for an array with length $n + 1$. Also index "-1" means the last.

Algorithm 2 DP2

```
OPT = [[−1, ∞] × (n + 1)]
OPT[n] = [−1, 0]
for i ∈ [n − 1, 0] do
    temp = g_i,    price = 0
    for j ∈ [i + 1, n] do
        temp = temp + g_j,    price = price + g_j · (j − i)
        if temp > L then
            break
        end if
        if OPT[i][1] > OPT[j][1] + price + P then
            OPT[i] = [j, OPT[j][1] + price + P]
        end if
    end for
end for
result = [[1, 0]],    k = OPT[0][0]
while k ≠ −1 do
    for i ∈ [k + 1, OPT[k][0]] do
        result[-1][0] = result[-1][0] + g_i
    end for
    result.append([k, 0])
    k = OPT[k + 1][0]
end while
return result;
```

1. Subproblem: OPT_i means the lowest total cost from day $i + 1$ to day n if we start with no gas in day $i + 1$. This means we must order gas in this day. Therefore if $i = 0$, then OPT_i is the total minimum cost for all the days. If $i = n$, then $OPT_i = 0$ because we do not need gas.

2. Recurrence: Since we should order $\sum_{i=1}^{n} g_i$ totally, this cost can be omitted because it is fixed except the delivery cost. Therefore we know $OPT_{n-1} = P$. Also notice that if $i + 1$ is the last day to order, then the total cost is P plus the store cost from day $i + 1$ to day n. In the same way, if $i + 1$ is not the last day to order and suppose the next is $j + 1$, then the total cost is P plus the store cost from day $i + 1$ to $j + 1$ plus OPT_j.

Therefore we prove the correctness. Notice we should not the number of an order be larger than L, which is considered in the algorithm. So we have the following relation formula:

$$OPT_i = P + \min_{i<j\le n}\left(\sum_{k=i+1}^{j} g_{j+1}\cdot(j-i) + OPT_j\right) \quad \text{s.t.} \quad \sum_{k=i+1}^{j+1} g_k \le L$$

At last, we output the result. Each member in array result is a 2-element array. The first element is the day to order and the second is the number to order.

3. Complexity: The time complexity is $O(n^2)$. The space complexity is $O(n)$.