Algorithm Design and Analysis (Fall 2023)

Assignment 4

Deadline: Dec 26, 2023

Yuqing Tu 522030910152

1. (30 points) Consider that you are in a stock market and you would like to maximize your profit. Suppose the prices of the stock for the n days, $p_1, p_2, \ldots, p_n$, are given to you. On the i-th day, you are allowed to do exactly one of the following operations:

   - Buy one unit of the stock and pay the price p_i. Your stock will increase by 1.
   - Sell one unit of stock and get the reward p_i if your stock is at least 1. Your stock will decrease by one.
   - Do nothing.

   Design an $O(n^2)$ time dynamic programming algorithm.

   Remark: [Not for credits] There exits a clever greedy algorithm that runs in $O(n \log n)$ time. Can you figure it out?

   Solution:

   Algorithm 1 Maximizing profit

   $dp[1][0] \leftarrow 0, dp[1][1] \leftarrow -p_1$
   $dp[1][i] \leftarrow -\infty$ for all i from 2 to n
   for all i from 1 to n **do**
       for all j from 0 to i **do**
           if $j = 0$ **then**
               $dp[i][j] \leftarrow \max\{dp[i-1][j], dp[i-1][j+1]\}$
           else if $j = n$ **then**
               $dp[i][j] \leftarrow \max\{dp[i-1][j-1], dp[i-1][j]\}$
           else
               $dp[i][j] \leftarrow \max\{dp[i-1][j-1], dp[i-1][j], dp[i-1][j+1]\}$
           end if
       end for
   end for
   return $\max\{dp[n][j]\}$ for all j from 0 to n

   Time complexity:
   There are two nested loops, the total number of cycles is $\sum_{i=1}^{n}(i+1) = \frac{(n-1)(n+2)}{2}$ and each cycle's time complexity is $O(1)$. So the time complexity is $O(n^2)$

2. (30 points) Given two strings $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$, we wish to find the length of their *longest common subsequence*, that is, the largest k for which there are indices $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ with $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{j_1} y_{j_2} \cdots y_{j_k}$. Design an $O(n^2)$ dynamic programming algorithm for this problem.

Solution:

Algorithm 2 The longest common subsequence

```
dp[0][j] ← 0 for all j from 0 to n
dp[i][0] ← 0 for all i from 0 to n
max ← 0
for all i from 1 to n do
    for all j from 1 to n do
        if x_i = y_j then
            dp[i][j] ← dp[i − 1][j − 1] + 1
        else
            dp[i][j] ← 0
        end if
        if max < dp[i][j] then
            max ← dp[i][j]
        end if
    end for
end for
return max
```

Time complexity:

There are two nested loops, the total number of cycles is n^2 and each cycle's time complexity is $O(1)$. So the time complexity is $O(n^2)$

3. (40 points) In the *Traveling Salesman Problem* (TSP), we are given an undirected weighted complete graph $G = (V, E, w)$ (where $(i, j) \in E$ for any $i \neq j \in V$). The objective is to find a cycle of length $|V|$ with minimum total weight, i.e., to find a tour that visit each vertex exactly once such that the total distance traveled in the tour is minimized. Obviously, the naïve exhaustive search algorithm requires $O((n-1)!)$ time. In this question, you are to design a dynamic programming algorithm for the TSP problem with time complexity $O(n^2 \cdot 2^n)$.

   (a) (10 points) Show that $n^2 \cdot 2^n = o((n-1)!)$, so that the above-mentioned algorithm is indeed faster than the naïve exhaustive search algorithm.

   (b) (30 points) Design this algorithm. Hint: label all vertices as $1, 2, \ldots, n$; given $i \in V$ and $S \subseteq V \setminus \{1, i\}$, let $d(S, i)$ be the length of the shortest path from 1 to i where the intermediate vertices are *exactly* those in S; show that the minimum weight cycle/tour is $\min_{i=2,3,\ldots,n}\{d(V \setminus \{1, i\}, i) + w(i, 1)\}$.

Solution:

(a)

$$\lim_{n\to\infty} \frac{n^2 \cdot 2^n}{(n-1)!} = 2 \cdot \lim_{n\to\infty} \frac{2n}{n-1} \cdot \lim_{n\to\infty} \frac{2n}{n-2} \cdot \lim_{n\to\infty} \prod_{i=1}^{n-3} \frac{2}{i} = 2 \cdot 2 \cdot 2 \cdot 0 = 0$$

So $n^2 \cdot 2^n = o((n-1)!)$

(b) **Algorithm 3** TSP

find the minimum path $l[u][v]$ between $u, v \in v$
$dp[\{u\}][u] \leftarrow l[u][1]$ for all $u \in V \backslash \{1\}$
for all S from binary of 1 to binary of 2^n **do**
    for all u from 1 to n **do**
        if $u \in S$ and $|S| \neq 1$ **then**
            $dp[S][u] = \min_{v \in S \backslash \{u\}}\{dp[S \backslash \{u\}][v] + l[v][u]\}$
        end if
    end for
end for
return $dp[V][1]$

Time complexity:
The time complexity of finding the minimum edges is $O(n^3)$ (use Dijkstra Algorithm). And there are two nested loops, the total number of cycles is $n \cdot 2^n$ and each cycle's time complexity is $O(n)$. So the time complexity is $O(n^2 \cdot 2^n)$ Then the total time complexity is $O(n^3 + n^2 \cdot 2^n) = O(n^2 \cdot 2^n)$.

4. How long does it take you to finish the assignment (including thinking and discussion)? Give a score (1,2,3,4,5) to the difficulty. Do you have any collaborators? Please write down their names here.

   It took me about 3 hours to finish the assignment. I'll give a 2 to the difficulty.