CS 101 Fall 2024 - Quiz 11
December 16, 2024 - 20 Minutes **Name:** **Student ID:**

---

1. (2 points) Honor Code

I promise that I will complete this quiz independently and will not use any electronic products or paper-based materials during the quiz, nor will I communicate with other students during this quiz.

I will not violate the Honor Code during this quiz.

$\checkmark$ True ○ False

2. (4 points) True or False

Determine whether the following statements are true or false.

(a) (1') The Floyd-Warshall algorithm can return the shortest path between all pairs of nodes in a connected graph with n nodes in $O(|V|^3)$, while it only needs $O(|V|^2)$ to find the shortest path between a given pair of nodes in the graph(for worst case). ○ True $\checkmark$ False

(b) (1') We can modify the Floyd-Warshall algorithm to detect whether there exists a negative cycle or not in a directed graph.

$\checkmark$ True ○ False

(c) (1') Since the solutions of DP problems depend on other subproblems, one needs to at least give out the solutions of some subproblems to solve the problem.

$\checkmark$ True ○ False

(d) (1') We could always use recursion in DP and hence do not need to allocate extra space to store solutions of subproblems, which could save the memory use. ○ True $\checkmark$ False

3. (4 points) Floyd-Warshall's Algorithm

Given the Floyd-Warshall's algorithm, Please write the fill in the blanks below. Write the most precise and simplified form.

Hint: *Mind the order of loop!*

Algorithm 1 Floyd-Warshall's algorithm

---

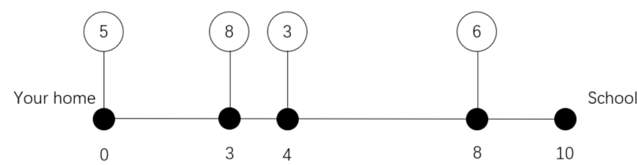
procedure FLOYD-WARSHALL(V, E)
    let *dist* be a $|V| \times |V|$ array of minimum distances initialized to ∞ (infinity)
    for each edge $(u, v) \in E$ **do**
        $dist[u][v] \leftarrow w(u, v)$ ▷ The weight of the edge (u, v)
    end for
    for each vertex $v \in V$ **do**
        $dist[v][v] \leftarrow 0$
    end for
    for i from 1 to $|V|$ **do**
        for j from 1 to $|V|$ **do**
            for k from 1 to $|V|$ **do**
                if dist[j][k] > dist[j][i] + dist[i][k] **then**
                    dist[j][k] ← dist[j][i] + dist[i][k]
                end if
            end for
        end for
    end for
    return *dist*
end procedure

---

4. (10 points) Remove the road sign

There is a road from your home to school. The length of the road is L kilometers, and your home is located at coordinate 0, and the school is located at coordinate L. There are n signs along the road, where the i 'th sign has a value a_i, indicating that you must travel at a speed of exactly a_i minutes per kilometer until you reach the next sign. There is also a sign at coordinate 0 which sets the initial speed. We can use the signs to calculate the time to travel from home to school. For example, in Fig. 1, the total time is $3 \times 5 + 1 \times 8 + 4 \times 3 + 2 \times 6 = 47$ minutes.

We now want to remove at most k signs in a way that minimizes the time it takes to travel from home to school. You cannot remove the sign at coordinate 0. We want to design a dynamic programming algorithm.



(a) (6') Define $OPT(i, j)$ be the minimum time needed to spend if we already walk to sign i, and choose j signs (i.e remove $i - j$ signs). Give your Bellman equation and explanation to solve the subproblems. (Write the Bellman Equation will receive 4 points, while the explanation will receive 2 points)

Solution:

$$OPT(i, j) = \begin{cases} 0 & i, j = 1 \\ \min_{l=1}^{i-1}\{OPT(l, j-1) + a_l(s_i - s_l)\} & \text{otherwise} \end{cases}$$

For all $1 \leq j \leq i \leq n$, s_i should be the distance from the start.

We can prove its correctness since every time we transfer the state from a lower sign to a higher sign, the problem is stated to "select enough signs and make total time minimized.

(b) (2') What is the answer to this question in terms of your subproblems?

Solution:

$$\min\{OPT(i, j) + a_i(L - s_i) \mid n - k \leq i \leq n, n - k \leq j \leq i\}$$

(c) (2') What is the runtime complexity of your algorithm?

Solution: Since we have total $\Theta(n^2)$ states, and each state needs $\Theta(n)$ time to transfer from. Hence it's $\Theta(n^3)$ time complexity.