Minimum Total Distance Traveled

solved by	Senan
Platform	LeetCode
difficulty	Hard
# Serial	2463
tags	Dynamic Programming
language	C++
solved on	@31/10/2024
link	https://leetcode.com/problems/minimum-total-distance-traveled/description/
Completion	☑

Intuition

The goal is to assign robots to factories such that the total distance they travel is minimized. By sorting both robots and factories based on their positions, the task simplifies as we can consider the closest available factories for each robot.

Approach

1. **Sorting**:
   - Sort both `robot` and `factory` by their positions to consider the closest options.
2. **Dynamic Programming**:
   - Define a DP table `dp[i][j]` where `i` represents the number of robots assigned so far and `j` represents the number of factories considered.
   - Initialize `dp[i][n]` to `LLONG_MAX` (infinity) for all `i < m` since no more assignments can be made when factories are exhausted.
3. **Deque Optimization**:
   - Use a deque to store valid states, allowing us to efficiently compute the minimum distance by tracking the prefix sum for each factory's assignments.
   - This helps us determine the minimum travel distance for each robot-factory combination by adding robots to factories until capacity is reached.
4. **Updating the DP Table**:
   - Traverse backwards over the robots and factories, updating `dp[i][j]` based on minimum distance calculations using prefix sums.
   - Use the deque to manage overlapping subproblems and efficiently find the minimum distance for each robot assignment.
5. **Result**:
   - The answer is stored in `dp[0][0]`, representing the minimum distance when starting from zero robots and factories.

Complexity

Time Complexity:

- Sorting the robots and factories takes **O(m log m + n log n)**.

- Filling the `dp` table involves iterating through robots and factories, making this part **O(mn)**.
- Each deque operation is efficient due to amortized constant time complexity.

Overall, the time complexity is **O(m log m + n log n + mn)**.

Space Complexity:

- The space complexity is **O(mn)** for the `dp` table and **O(m)** for the deque.

Code

```
class Solution {
public:
    long long minimumTotalDistance(vector<int>& robot, vector<vector<int>>& factory) {
        sort(robot.begin(), robot.end());
        sort(factory.begin(), factory.end());

        int m = robot.size();
        int n = factory.size();
        vector<vector<long long>> dp(m + 1, vector<long long>(n + 1, LLONG_MAX));

        for (int j = n - 1; j >= 0; j--) {
            long long prefix = 0;
            deque<pair<int, long long>> qq;
            qq.push_back({m, 0});

            for (int i = m - 1; i >= 0; i--) {
                prefix += abs(robot[i] - factory[j][0]);

                while (!qq.empty() && qq.front().first > i + factory[j][1]) {
                    qq.pop_front();
                }

                while (!qq.empty() && qq.back().second >= dp[i][j + 1] - prefix) {
                    qq.pop_back();
                }

                qq.push_back({i, dp[i][j + 1] - prefix});
                dp[i][j] = qq.front().second + prefix;
            }
        }
        return dp[0][0];
    }
};
```