Analysis of Algorithms and Data Structures: Problem set 2

Giosue Castellano

Problem 6

We can model this problem of going from the top left to the bottom right corner, collecting the maximum amount of diamonds, and then going back and collecting the maximum of leftover diamonds in the following way: We start with two paths from the top left corner and both try to maximize the number of diamonds on the way to the bottom right corner and as soon as one path collects a diamond, the other path cannot collect it anymore. This works specifically because on the way to the bottom we are restricted to going either right or down, and going up the directions are flipped, i.e. we can go either left or up.
We now obtain four parameters i_1, j_1, i_2, j_2, where i specifies the row of the respective path and j the respective column. We can exploit the fact that both paths have to be of the same length, because they both take 1 step at a time, in order to write one parameter as a sum of the others, I take j_2. With d being the total length of the path, we can get j_2 by this equation: $j_2 = d - i_2$, because $d = i_1 + j_1 = i_2 + j_2$. This ensures that we can achieve a running time of $O(n^3)$.
Now let $dp[i_1][j_1][i_2]$ be the maximum number of diamonds that can be collected when 2 paths are at position (i_1, j_1) and (i_2, j_2) respectively. Both paths can either take a step down or a step right at any given time.
The result will be $dp[n-1][n-1][n-1]$ and it is initialized to $dp[0][0][0] = b[0,0]$, where b is an $n \times n$ matrix that keeps track of where the diamonds are in the grid.
I will define the function $sum(i_1, j_1, i_2, j_2)$ as the number of diamonds the path can pick up when they go the given parameters. Note that this can be at most 2 in the case they both find separate diamonds. If they find the same diamond, it will only count once to the total count since only one path can pick it up.

$$\text{sum}(i_1, j_1, i_2, j_2) = \begin{cases} b[i_1, j_1] + b[i_2, j_2] & \text{if } (i_1, j_1) \neq (i_2, j_2) \\ b[i_1, j_1] & \text{if } (i_1, j_1) = (i_2, j_2) \end{cases}$$

Here is the pseudocode for this problem with a running time of $O(n^3)$. We iterate over all lengths and respective positions to find the maximum number of diamonds. With the four equations, we cover all possible scenarios of the movement of both paths and in the end can recover the solution:

```
for d in range(2n - 2, -1, -1):
    for i_1 in range(min(n - 1, d)):
        for i_2 in range(min(n - 1, d)):
            j_1 = d - i_1
            j_2 = d - i_2
            dp[i_1][j_1][i_2] = max(
                dp[i_1 + 1][j_1][i_2 + 1] + sum(i_1, j_1, i_2, j_2),
                dp[i_1][j_1 + 1][i_2] + sum(i_1, j_1, i_2, j_2),
                dp[i_1][j_1 + 1][i_2 + 1] + sum(i_1, j_1, i_2, j_2),
                dp[i_1 + 1][j_1][i_2] + sum(i_1, j_1, i_2, j_2)
            )
```