Problem set 1

Federico Chinello

January 10, 2024

I assume array and matrix indices start from 1. When I do slicing, I assume both the extremes are selected (i.e., `A[i:j]` comprises both `A[i]` and `A[j]`).

Problem 7

Each field of a square $n \times n$ board either contains a single diamond or it is empty. We start in the top left corner of the board, and take steps either down or to the right, until we reach the bottom right corner. Compute the largest number of diamonds we can collect along the way. Running time: $O(n^2)$.

We represent the board as a n x n matrix `B[][]`. `B[i][j]` is `1` if the filed contains a diamond and `0` if it is empty.

```
MAXDIAMONDS(B[][]):

    // dp[i][j] is the maximum number of diamonds
    // I collect if I reach B[i][j] from B[1][1]
    // following the optimal path
    dp[][] := n x n matrix
    // Base case:
    dp[1][1] = B[1][1]
    // Recursive cases:
    for i = 2,...,n:
        dp[i][1] = dp[i-1][1] + B[i][1]
    for j = 2,...,n:
        dp[1][j] = dp[1][j-1] + B[1][j]
    for i = 2,...,n:
        for j = 2,...,n:
            dp[i][j] = max{dp[i-1][j], dp[i][j-1]} + B[i][j]

    return B[n][n]
```