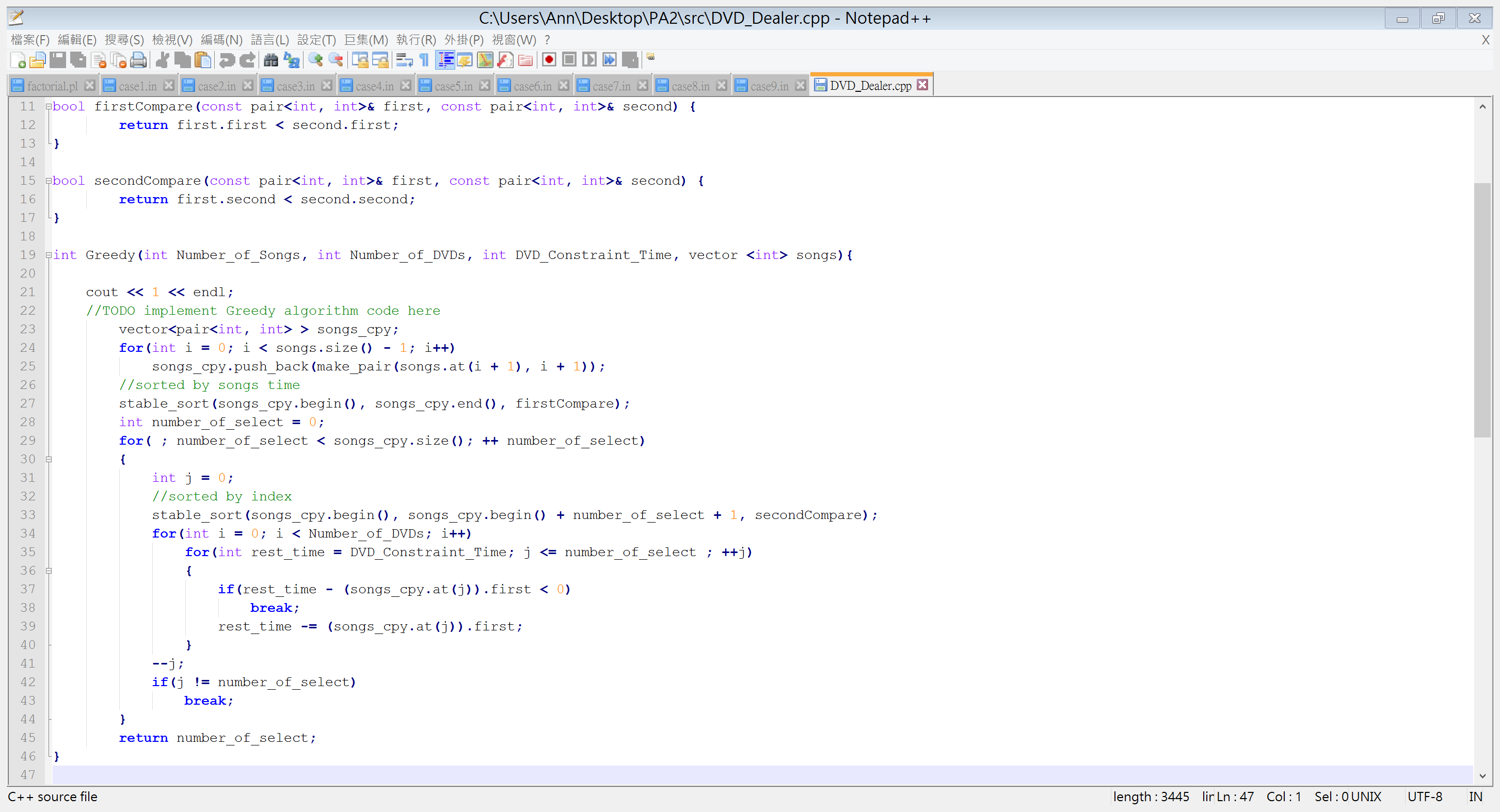
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| case | GD | | | DP | | |
|  | Maximum songs | CPU time(s) | Memory(MB) | Maximum songs | CPU time(s) | Memory(MB) |
| case1  N=4 | 1 | 0 | 11.896 | 1 | 0 | 11.896 |
| case2  N=1 | 0 | 0 | 11.896 | 0 | 0 | 11.896 |
| case3  N=15 | 0 | 0 | 11.896 | 0 | 0 | 11.896 |
| case4  N=1000 | 280 | 0.001999 | 11.896 | 285 | 0.058991 | 17.084 |
| case5  N=100000 | 10133 | 1.65775 | 14.496 | 10265 | 111.243 | 14418.3 |
| case6  N=100000 | 12882 | 2.51262 | 14.496 | 13102 | 186.073 | 22704.3 |
| case7  N=1000 | 1000 | 0.022996 | 11.896 | 1000 | 0.294955 | 160.768 |
| case8  N=10 | 0 | 0 | 11.896 | 0 | 0.000999 | 12.292 |
| case9  N=100 | 61 | 1 | 11.896 | 62 | 0.003999 | 12.292 |

**Programming assignment #2 DVD Dealer**  b03901155黃安

1. Greedy algorithm: complexity analysis



line24~25: n – 1

line27: nlgn

line28: 1

line31: n

line33: 1lg1 + 2lg2 +…+nlgn

line34: (multiple line35~40 by c)

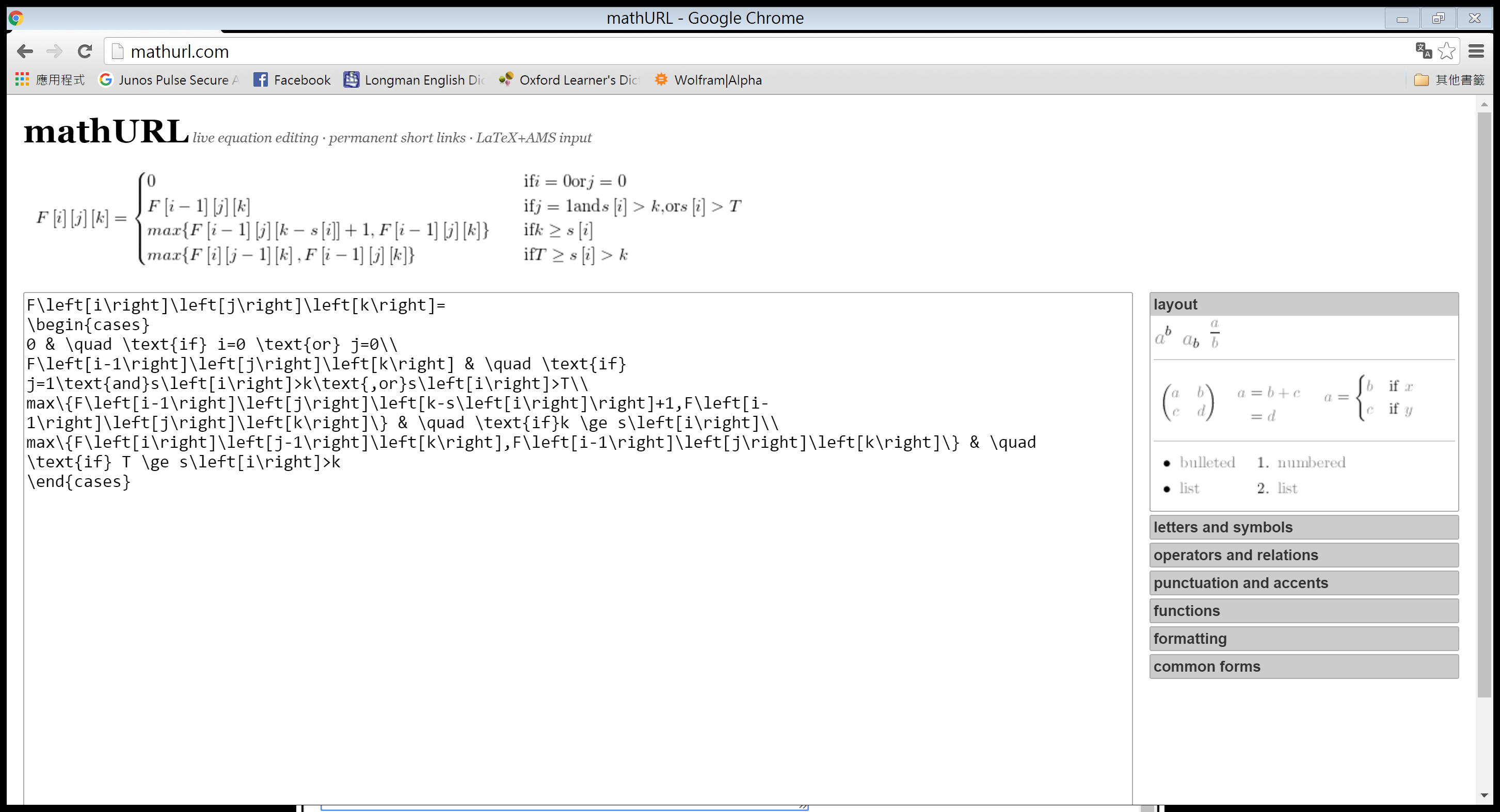
line35~40: c(1^2 + 2^2 +⋯+ n^2)

line41: 1 + 2 +…+ n

line42~43: 1 + 2 +…+ n

line45: 1

Sum up: (n – 1) + (nlgn) + (1) + (n) + (1lg1 + 2lg2 +…+ nlgn) + c(1^2 + 2^2 +…+ n^2) + (1 + 2 +…+ n) + (1 + 2 +…+ n) + (1) = O(1^2 + 2^2 +…+ n^2)

1. Mathematical analysis

Assume F[i][j][k] is optimal. If F[i-1][j][k] isn’t optimal, we will replace it with another optimal solution by our assumption about F[i][j][k]. in addition, F[i-1][j][k-s[i]], and F[i][j-1][k] will be optimal for the same reason. As a result, every elements in F[i][j][k] will become optimal after we replace nonoptimal elements with optimal ones. In conclusion, this problem has optimal substructure, and we can use dynamic programming to solve it.

1. Dynamic programming: complexity analysis

What the dynamic programming does is find out every elements in F[N][M][T]. The time complexity of filling one element in the 3D array = O(1). Thus, filling the entire array needs O(NMT).