Student Name: Yinsheng Dong

Student Number: 11148648

NSID: yid164

CMPT360 Assignment 3

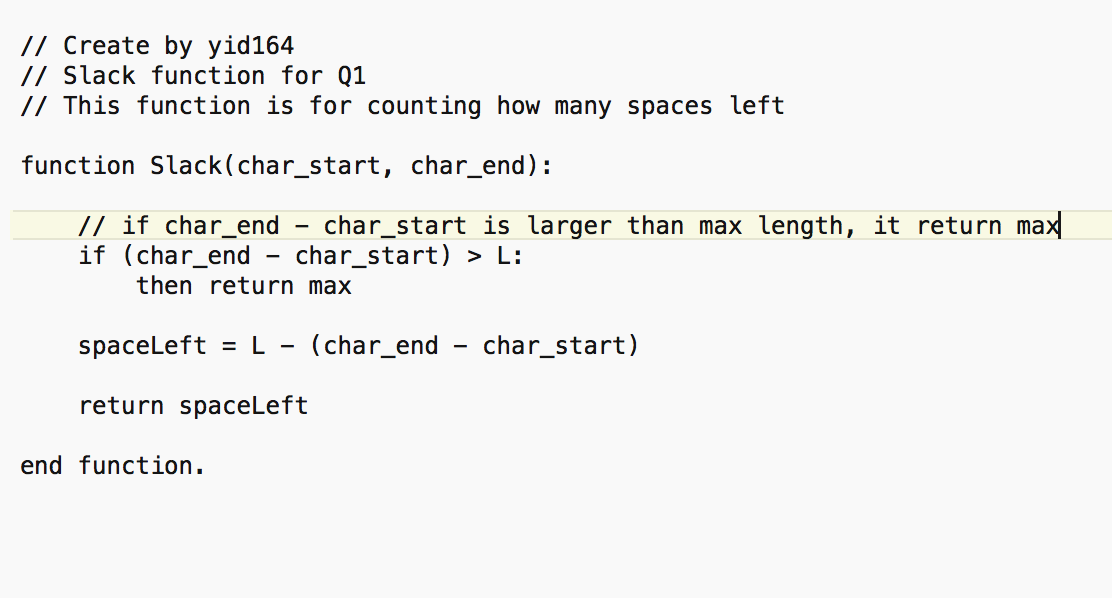
1. Kleinberg and Tardos p. 317 #6

The Good Printing Question.

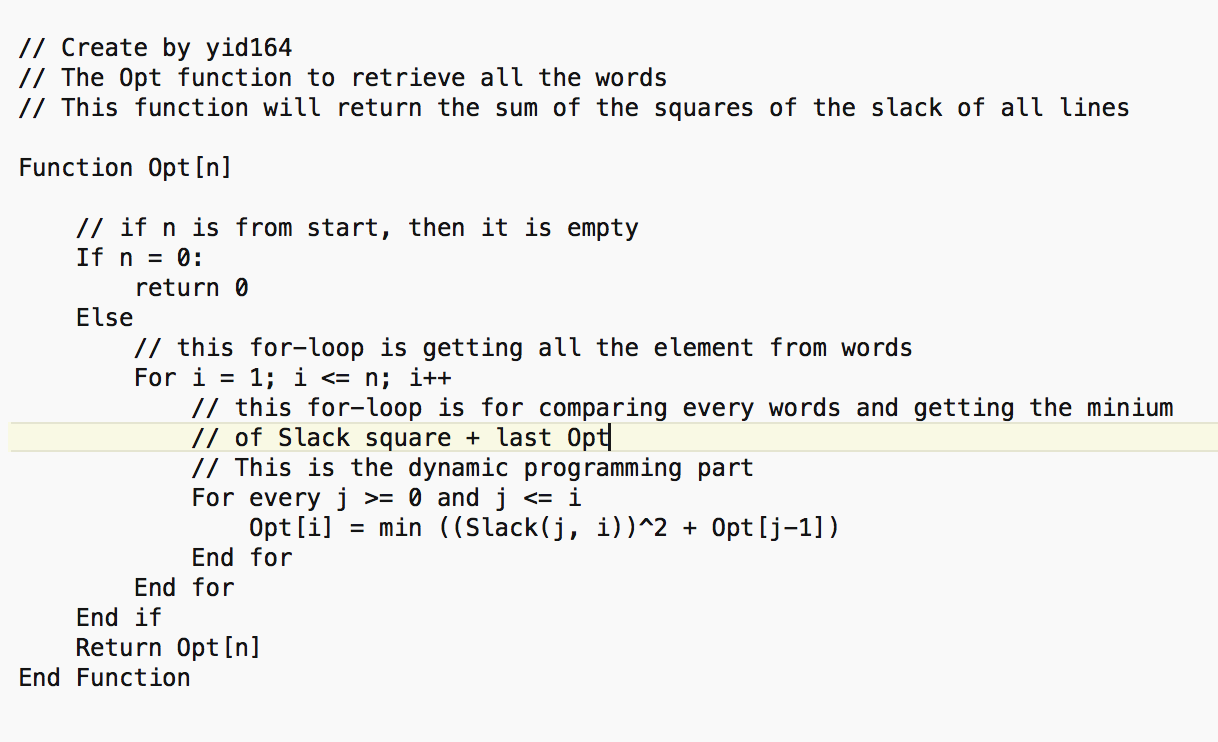
Solution:

Assume that we have W words for good printing, and L for the maximum line length. The question has given the space after each word except the last, and c as the characters for each word, and the slack is the number of spaces left at the right margin.

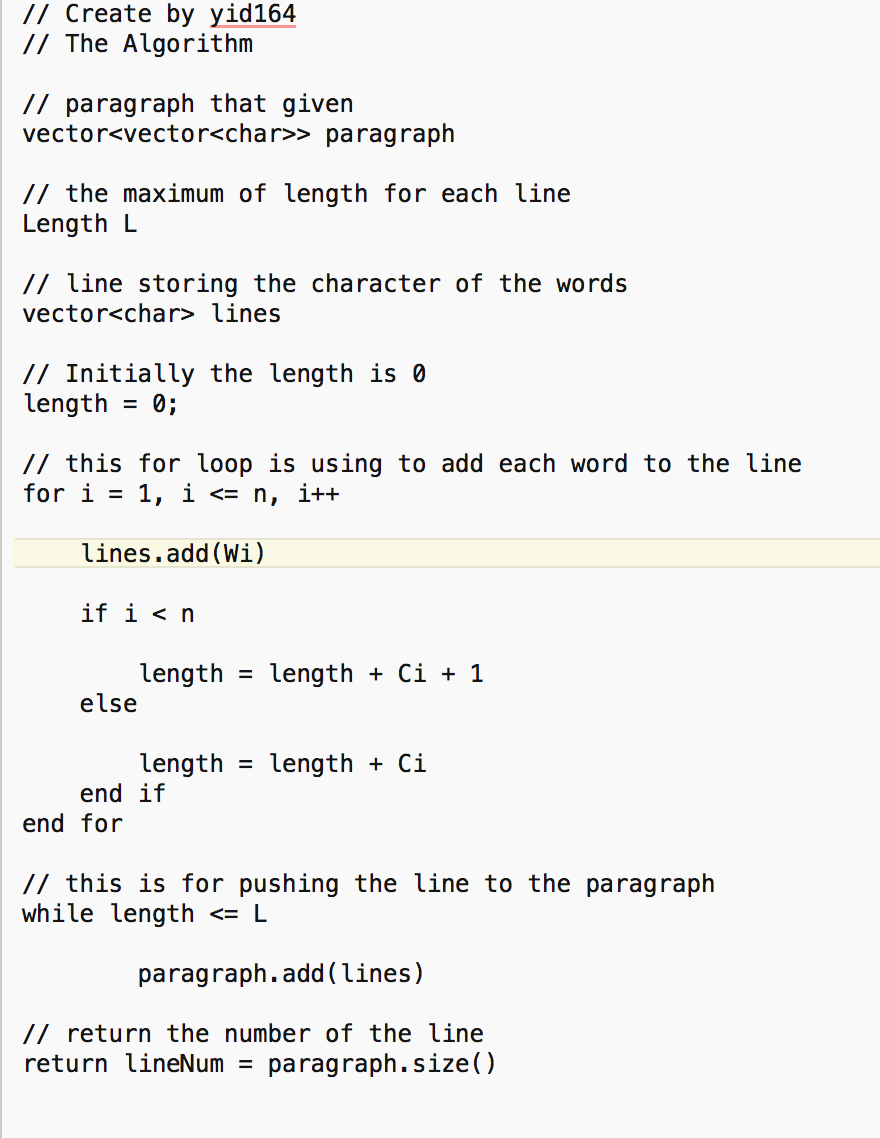
Firstly, we need to define the slack S. The slack function is for calculate how much space left when a word added.



Then, we define an optimum solution to retrieve all the words and to find the minimum one. We start at the bottom line because the last word will contain the space value.



The actual design algorithm from all above idea is:



In this solution, the Slack() function’s time complexity is O(n), but the Opt() function’s complexity is O(n^2), so the total time complexity is O(n^2)

1. Kleinberg and Tardos p. 323 # 11

The manufactures problem

Solution:

Company A charges a fixed rate r per pound: A = rsi

Company B charges every week: B = c

Let Opt(i) be the cost for the optimal schedule from weeks 1 to i

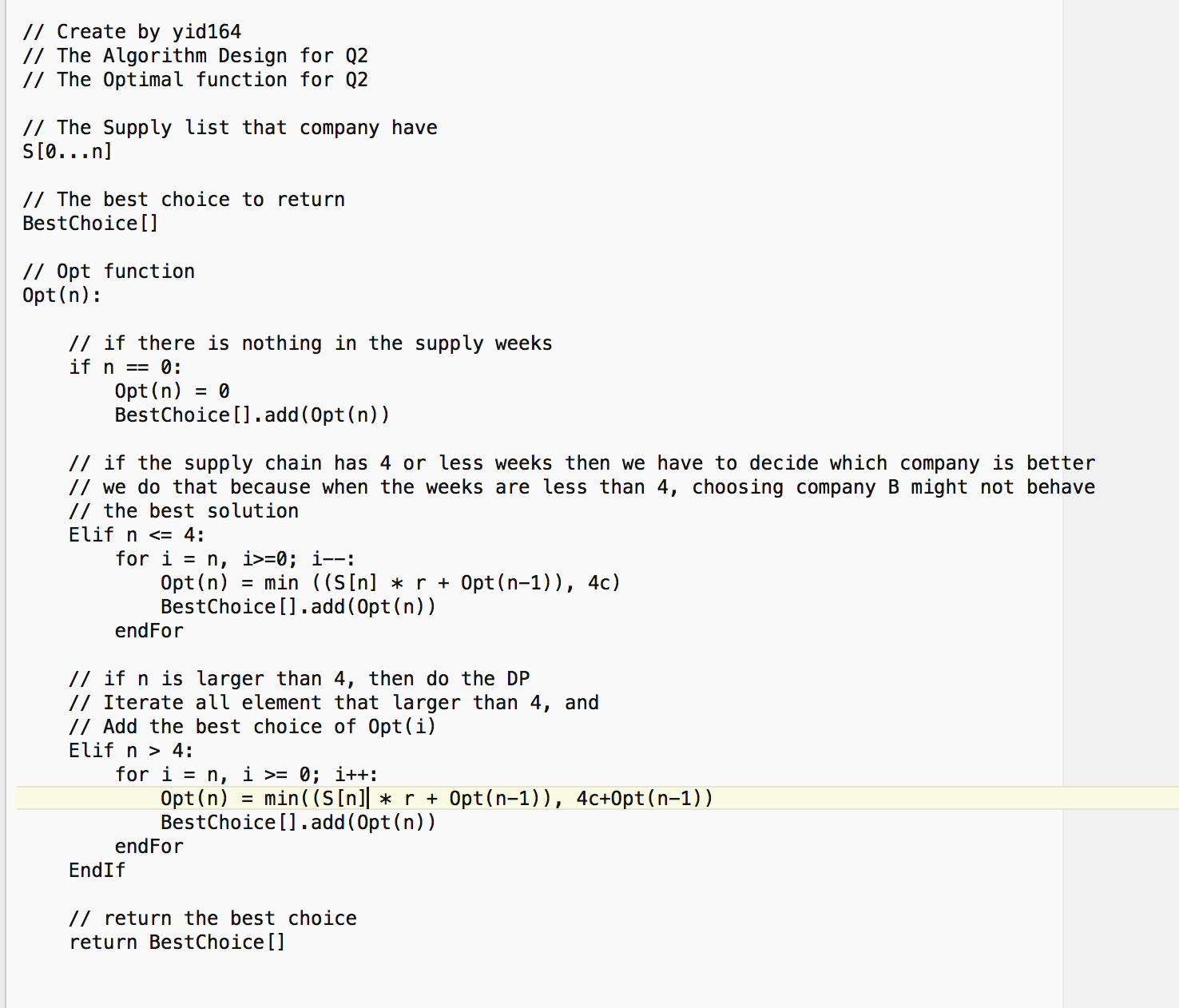
Company A: Opt(i) = r \* si + Opt(i-1)

Company B: Opt(i) = 4 \* c + Opt(i-4)

Then we need to determine which one is smaller when we choose combine company A and company B. However, if we choose company B and the i is smaller than 4, we won’t get the maximum benefit. In this case, we have to restrict the i is smaller or equal to i.

Since Company B iterates the pointer after 4 point, so we have to know the case i = 0, 1, 2, 3, and 4, then we can use the Opt(i) = min ((r \* s + Opt(i-1)), (4 \* c + Opt(i-4)).

Then, the algorithm is:



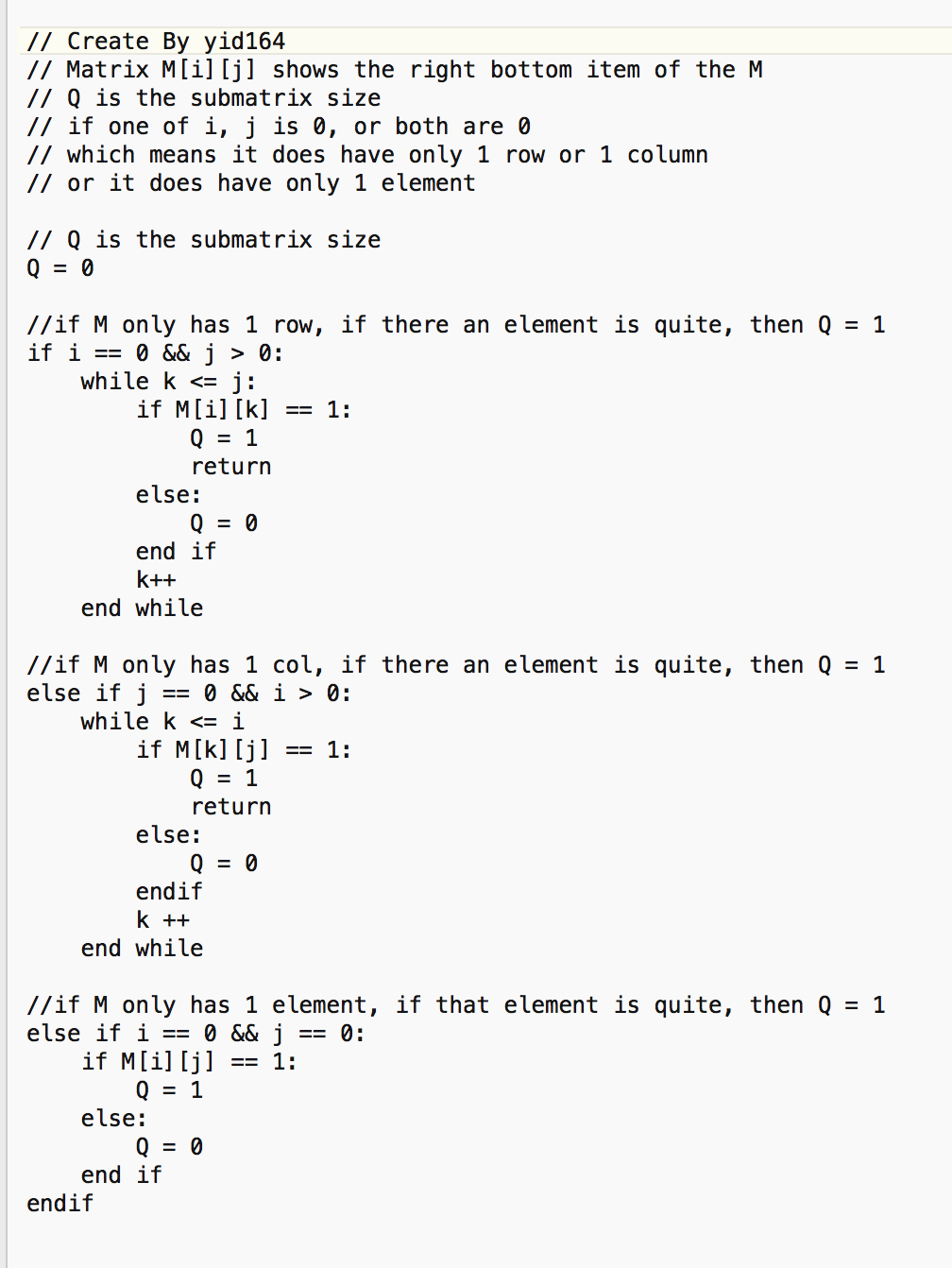
In this algorithm, the first if statement contains O(1) complexity, and the second if statement contains O(n^2) complexity, and the finally if statement is O(n^2) complexity. Then the total time complexity is O(n^2).

1. The matrix problem

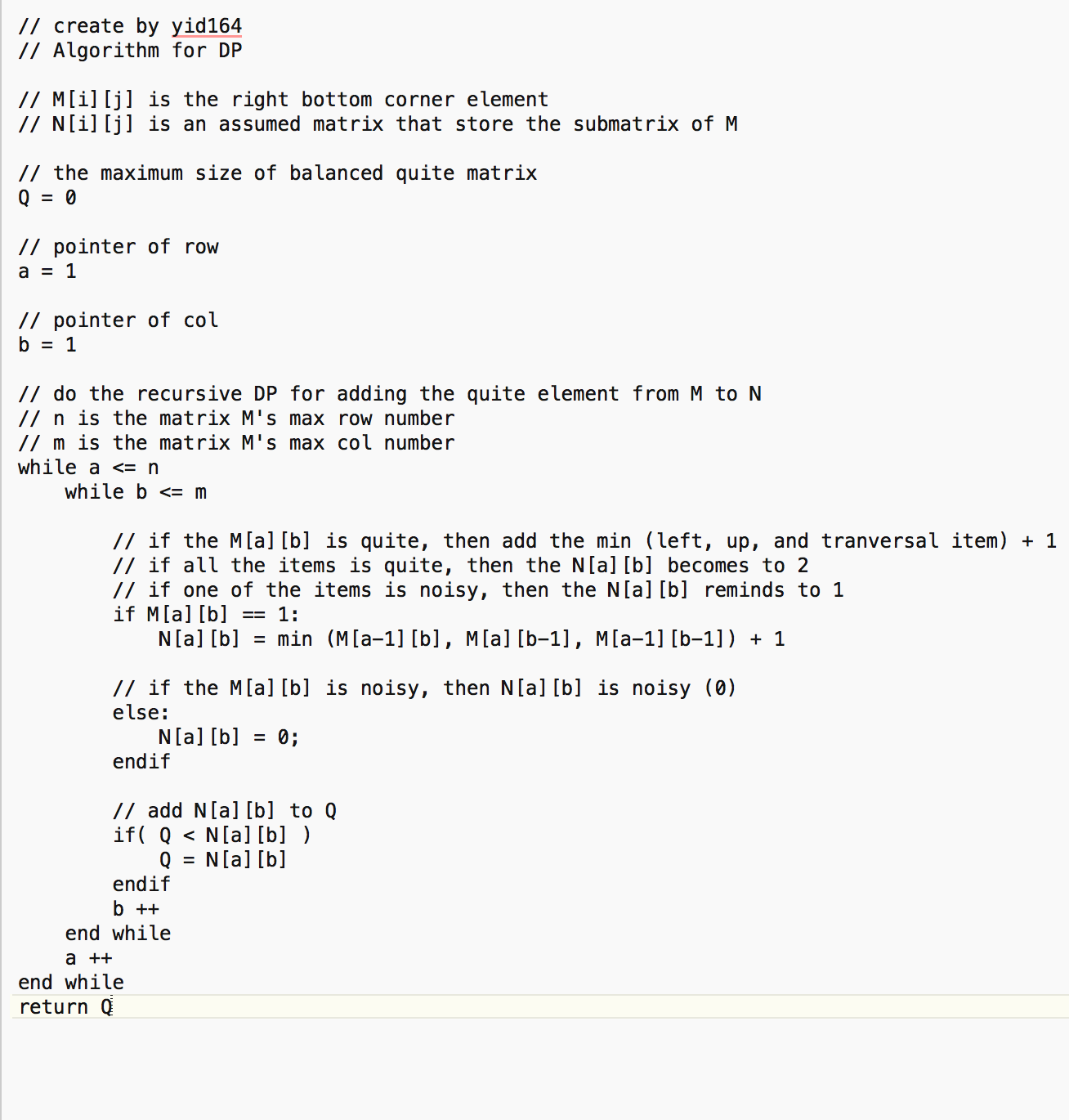
We assume that there is a new matrix N which is the largest balanced quite submatrix in M. The size of matrix N initially is same as the matrix M (which means for all M [i] [j] = ‘q’). Then we need to check every item in M to know which one is ‘q’, and the best way to start is from the right bottom corner. Also, we need to fit the N[i][j] which 1<= i <= n and 1<= j <= m, which n is the rows of M and m is the columns of M.

For convenience in this question, we assume that the quite = 1, and the noisy = 0.

First, we need to set the boundary condition when the M[i][j] size is 1, which means the matrix M only have 1 item, 1 row or 1 col. In next page



Then, we can design the algorithm:



This algorithm retrieves all the element from M(1,1) to the last item, and to adjust all the items recursively, then get the maximum of the quite balanced submatrix of M.

The time complexity of Algorithm 1 is O(2n), and the DP is O(n^2). Then the totally time complexity is O(n^2).