1. Knapsack Program

for w = 0 to W

B[0,w] = 0

for i = 1 to n

B[i,0] = 0

for i = 1 to n

for w = 0 to W

if wi <= w // item i can be part of the solution

if bi + B[i-1,w-wi] > B[i-1,w]

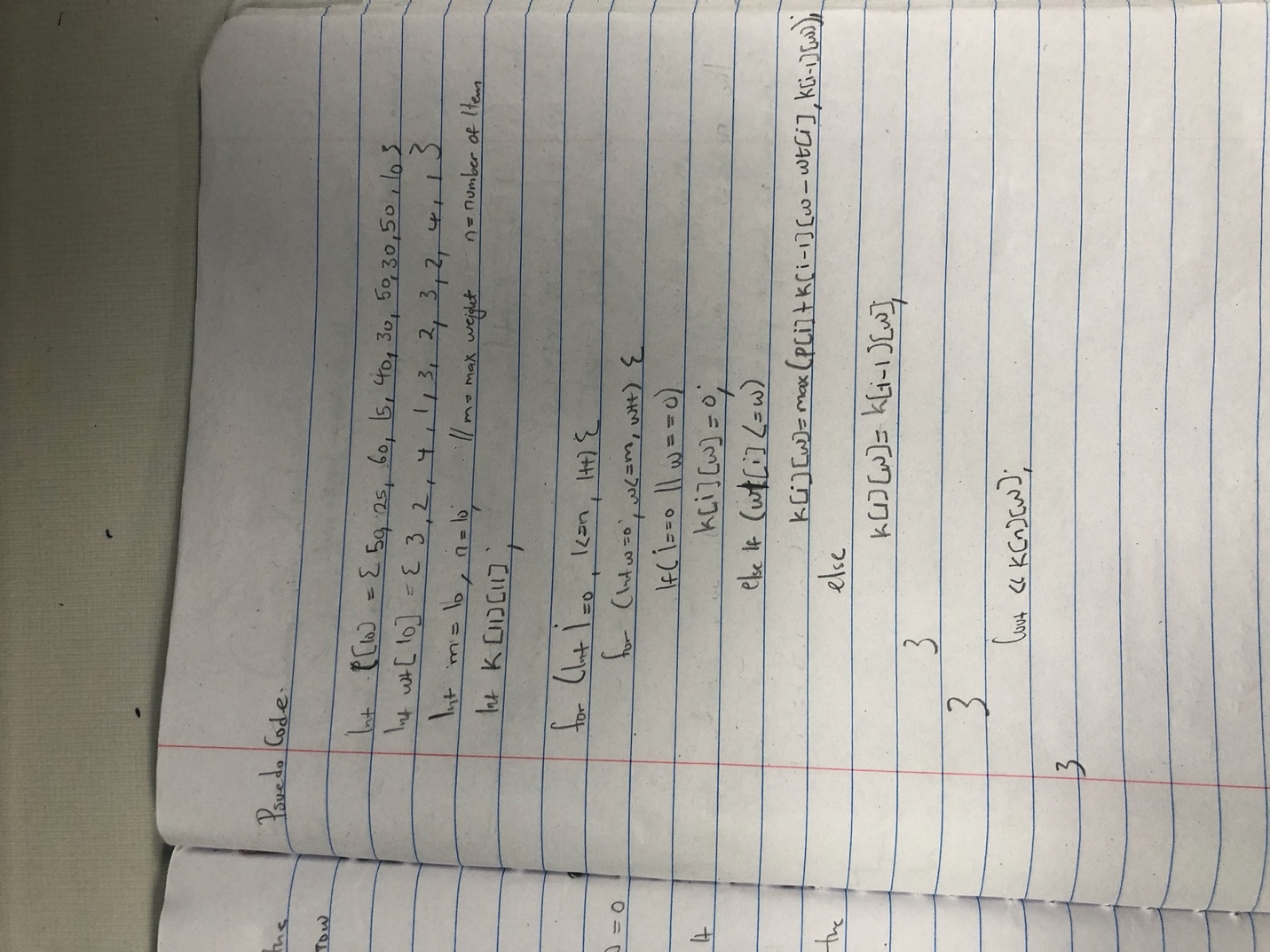
B[i,w] = bi + B[i-1,w- wi]

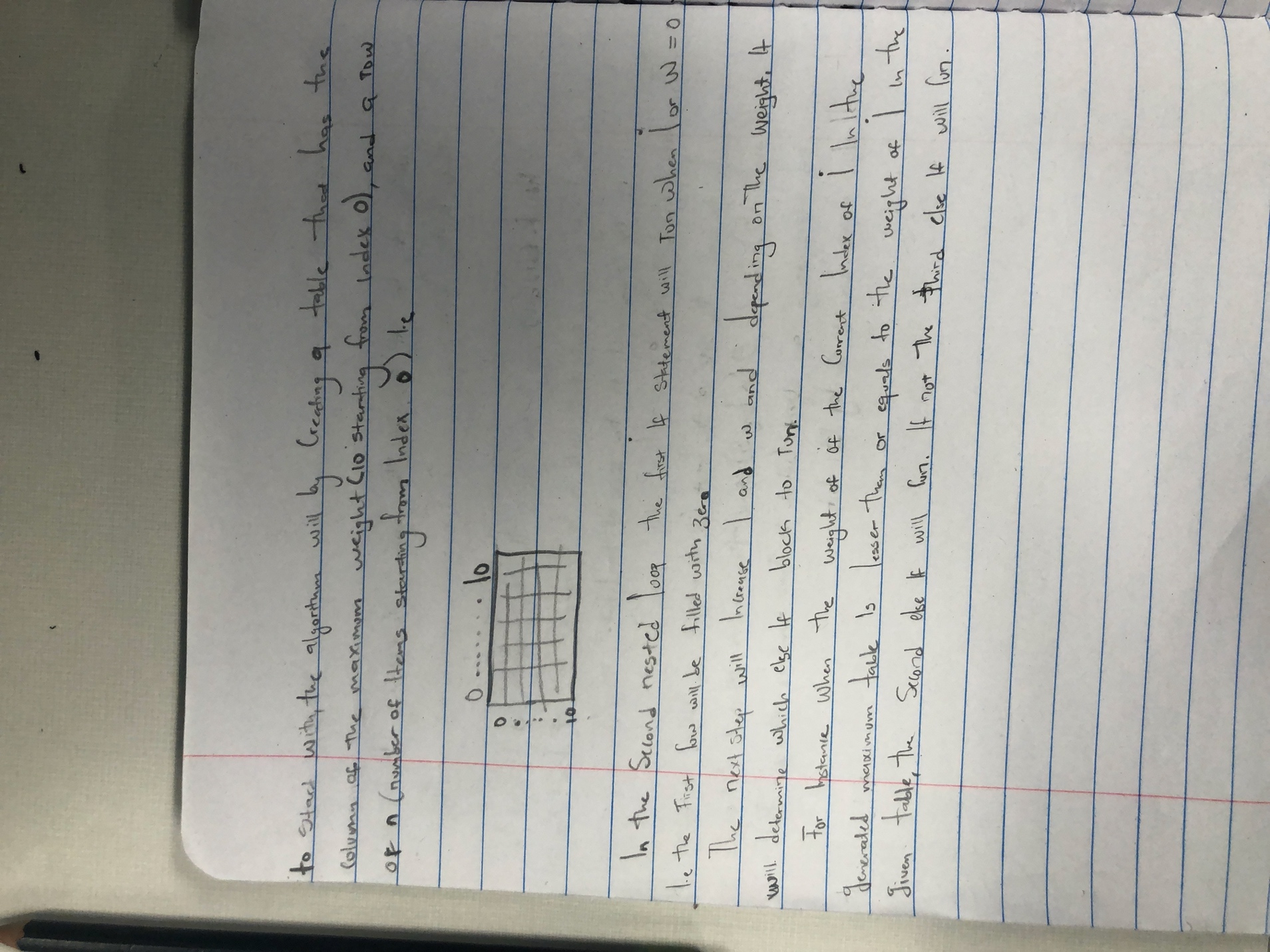
else

B[i,w] = B[i-1,w]

else B[i,w] = B[i-1,w] // wi > w}

b.





c.

A picture containing indoor

Description automatically generated

d.

for w = 0 to W

B[0,w] = 0 **O(W)**

for i = 1 to n

B[i,0] = 0

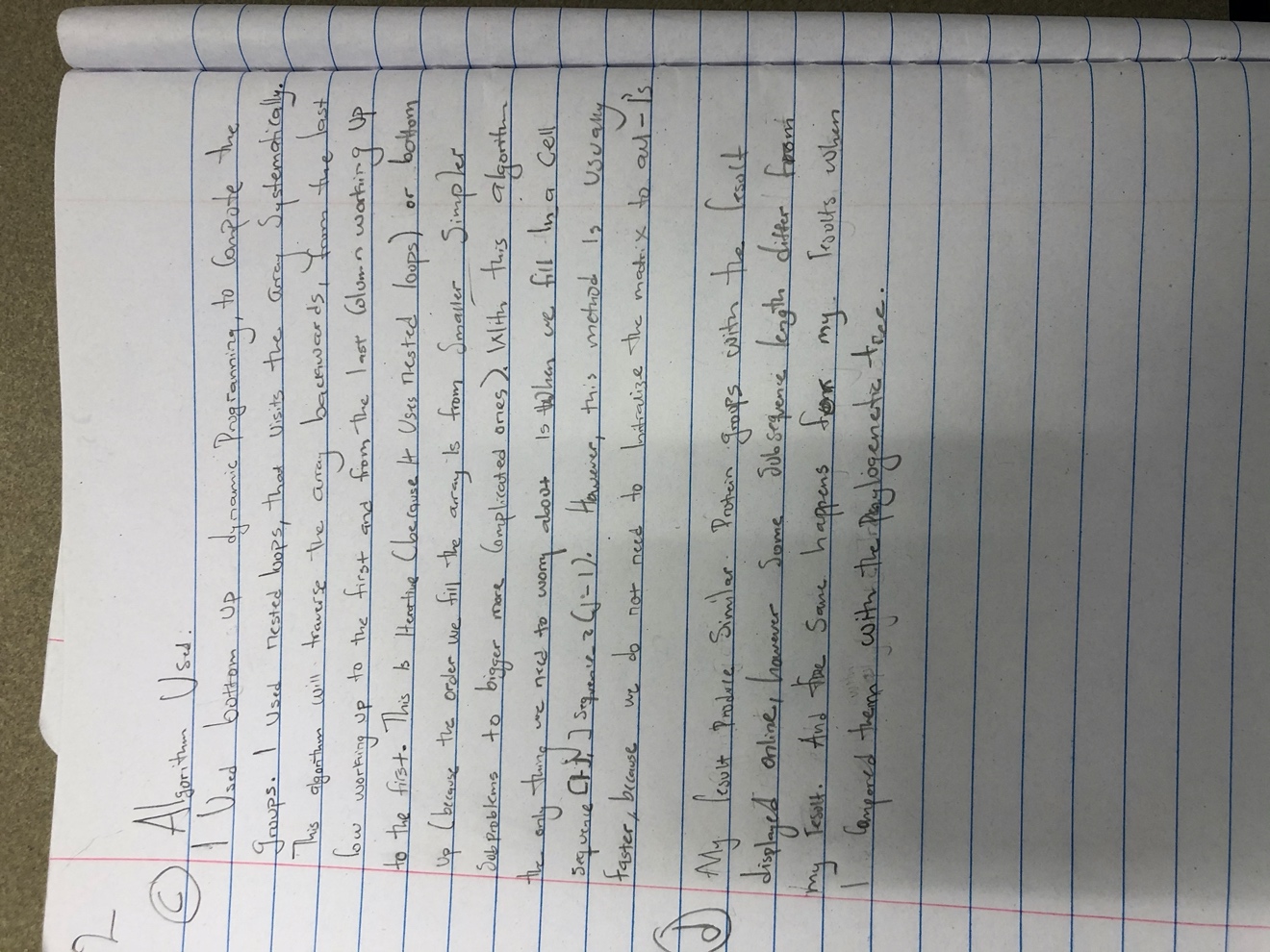
for i = 1 to n repeat n times

for w = 0 to W **O(W)**

< the rest of the code >

Running time is **O(n\*W)**

**2.**

****

3.

(a) We will use the Four Russian Algorithm, the best algorithm to give us the best space and time complexity. Let D be the dynamic programming table that is ﬁlled during the edit distance algorithm. The standard edit distance algorithm ﬁlls this table D row by row after initialization of the ﬁrst row and the ﬁrst column. Without loss of generality, we assume that all the edit operations cost unit time each. The basic idea behind the Four Russian Algorithm is to partition the dynamic programming table D into small blocks each of width and height equal to t where t is a parameter to be ﬁxed in the analysis. Each such block is called a t-block. The dynamic programming table is divided into t-blocks such that any two adjacent t-blocks overlap by either a row or column of width (or height) equal to t. After this partitioning is done The Four Russian algorithm ﬁlls up the table D block by block.

A quick qualitative analysis of the algorithm is as follows. After the partitioning of the dynamic programming table D into t-blocks we have nt2 blocks and if processing of each of the block takes O(t) time then the running time is O( nt ). In the case of standard dynamic programming, entries are ﬁlled one at a time (rather than one block at a time). Each entry can be ﬁlled in O(1) time and hence the total run time is O(n2 ). In the Four Russian algorithm, there are nt2

blocks. In order to be able to ﬁll each block in O(t) time, some preprocessing is

done Computing edit script in O(n2 / log n) Time and O(n) Space

D[i − 1, j − 1],D[i − 1, j] or D[i, j − 1]. D[i, j] ≥ D[i − 1, j − 1](characters at

S1 [i] and S2 [j] may be same or diﬀerent), D[i, j] ≤ D[i, j − 1] + 1 (cost of insert

is unity),D[i, j − 1] ≤ D[i − 1, j − 1] + 1(same inequality as the previous one

rewritten for element D[i, j − 1]). The following inequalities can be derived from

the previous inequalities.

−D[i, j] ≤ −D[i − 1, j − 1]

D[i, j − 1] ≤ D[i − 1, j − 1] + 1

−D[i, j] + D[i, j − 1] ≤ 1

D[i, j − 1] − D[i, j] ≤ 1

D[i, j] ≤ D[i, j − 1] + 1 {Started with this}

−1 ≥ D[i, j − 1] − D[i, j]

|D[i, j − 1] − D[i, j]| ≤ 1

Along the same lines we can also prove that |D[i − 1, j] − D[i, j]| ≤ 1 and

D[i − 1, j − 1] ≤ D[i, j].

(b) For the given root word and the candidates, we can put them in the place of the strings D1[i – 1, j -1], and D[i – 1, j]to obtain the minimum number of operations that are needed to obtain.

(c) There are also other factors to judge the most appropriate word other than the minimum number of changes.

1. INDEXING TOKENS is also one very important criteria to judge the correctly spelt word

2. RELATIVE POSITIONING is another important factor. It is the positioning of the letters in the typed word and the actual word. If more matching occurs, then it would probably be the actual word.

4. Discussion from the histogram of each novel with different language.

For each novel in English, the Huffman coding differs in length, because of the number of times it repeats in the novel. The same applies to the novel in other languages too, length of the Huffman code per letter differs because of the frequency and position in the file.