15.1 Another Example: Inverse Word Search

Let's flip the classic word search problem and instead create the board that contains the specified words! We'll be given the grid dimensions and the set of words, each of which must appear in the grid, in a straight line. The words may go forwards, backwards, up, down, or along any diagonal. Each grid cell will be assigned one of the 26 lowercase letters. We may also be given a set of words that should not appear anywhere in the grid. Here's an example:

Input:

4 4
+ arts
+ arid
+ east
+ rest
- ear
- at
- sit

e
r
a
a
e
r
a
a
s
s
i
t
t
t
t
t
d
d
s
s
i
i

e
b
a
o
e
h
k
c
a
p
o
o

In the middle above, is an incorrect solution. Though it contains the 4 required words, it also contains two of the forbidden words. The solution on the right is a fully correct solution. This particular problem has 8 total solutions including rotations and reflections.

Here's another example:

5 3
+ echo
+ baker
+ apt
+ toe
+ ore
+ eat
+ cap
And a couple more puzzles:

<table>
<thead>
<tr>
<th>3 3</th>
<th>7 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ ale</td>
<td>+ avocado</td>
</tr>
<tr>
<td>+ oat</td>
<td>+ magnet</td>
</tr>
<tr>
<td>+ zed</td>
<td>+ cedar</td>
</tr>
<tr>
<td>+ old</td>
<td>+ robin</td>
</tr>
<tr>
<td>+ zoo</td>
<td>+ chaos</td>
</tr>
<tr>
<td></td>
<td>+ buffalo</td>
</tr>
<tr>
<td></td>
<td>+ trade</td>
</tr>
<tr>
<td></td>
<td>+ lad</td>
</tr>
<tr>
<td></td>
<td>+ fun</td>
</tr>
<tr>
<td></td>
<td>- ace</td>
</tr>
<tr>
<td></td>
<td>- coat</td>
</tr>
</tbody>
</table>

15.2 Generating Ideas

- If running time & memory are not primary concerns, and the problems are small, what is the simplest strategy to make sure all solutions are found. Can you write a simple program that tries all possibilities?

- What variables will control the running time & memory use of this program? What is the order notation in terms of these variables for running time & memory use?

- What incremental (baby step) improvements can be made to the naive program? How will the order notation be improved?

15.3 Mapping Ideas to Code

- What are the key steps to solving this problem? How can these steps be organized into functions and flow of control for the main function?

- What information do we need to store? What C++ or STL data types might be helpful? What new classes might we want to implement?

15.4 Getting the Details Right

- What are the simplest test cases we can start with (to make sure the control flow is correct)?

- What are some specific (simple) corner test cases we should write so we won’t be surprised when we move to bigger test cases?

- What are the limitations of our approach? Are there certain test cases we won’t handle correctly?

- What is the maximum test case that can be handled in a reasonable amount of time? How can we measure the performance of our algorithm & implementation?