Project 2: Dynamic programming

COT 4400, Fall 2021

Due November 15, 2021

1 Overview

For this project, you will develop an algorithm to compute the maximum alignment between a target sequence and any possible combination of two given subsequences. Designing and implementing this solution will require you to model the problem using dynamic programming, then understand and implement your model.

You are only allowed to consult the class slides, the textbook, the TAs, and the professor. In particular, you are not allowed to use the Internet. This is a group project. The only people you can work with on this project are your group members. This policy is strictly enforced.

In addition to the group submission, you will also evaluate your teammates' cooperation and contribution. These evaluations will form a major part of your grade on this project, so be sure that you respond to messages promptly, communicate effectively, and contribute substantially to your group's solution. Details for your team evaluations are in Section 6.2. You will submit the peer evaluations to another assignment on Canvas, labelled "Project 2 (individual)."

A word of warning: this project is team-based, but it is quite extensive and a nontrivial task. You are highly encouraged to start working on (and start asking questions about) this project early; teams who wait to start until the week before the due date may find themselves unable to complete it in time.

2 Problem Description

In this problem, you are given three arrays, *seq1*, *seq2*, and *target*, and the solution to this problem will be the maximum alignment score s of *target* and a sequence *soln*, where *soln* is formed by combining the values in *seq1* and *seq2* in some order (i.e., *seq1* and *seq2* are subsequences of *soln*). Moreover, s will be the maximum alignment score of *target* with any combination of *seq1* and *seq2*.

For this project, we will calculate alignment scores as the unnormalized cosine similarity. In general, we compute cosine similarity of two normalized vectors or sequences $\vec{a}$ and $\vec{b}$ as a sum-product (or dot product):

$$\langle \vec{a}, \vec{b} \rangle = \sum_{i=1}^{n} a_i \cdot b_i,$$

where n is the length of $\vec{a}$ and $\vec{b}$, and a_i and b_i represent element i in $\vec{a}$ and $\vec{b}$, respectively. This score is only referred to as cosine similarity when the sequences are *normalized*; that is, they have the property that $\langle \vec{a}, \vec{a} \rangle = 1$; however, we will calculate the alignment score using the dot product even if the sequences involved are not normalized. That said, you may assume that the input sequences you are given for this problem will be designed so that *target* and any combination of *seq1* and *seq2* will be normalized.

3 Worked example

Consider the problem with *seq1* = (1, 2, 7), *seq2* = (1, 3, 6), and *target* = (1, 2, 1, 3, 7, 6). **Note:** after working through this example, we will discuss what a normalized version of this problem would look like.

The solution to this problem will be a score of 100, achieved by the sequence (1, 2, 1, 3, 7, 6) (*seq1* and *seq2* highlighted in red and blue, respectively). If we compute the dot product of this sequence with *target*, we see that it has an alignment score of:

$$\begin{aligned}\langle(1,2,1,3,7,6),(1,2,1,3,7,6)\rangle &= \sum_{i=1}^{6} a_i \cdot b_i \\ &= 1(1) + 2(2) + 1(1) + 3(3) + 7(7) + 6(6) \\ &= 1 + 4 + 1 + 9 + 49 + 36 \\ &= 100\end{aligned}$$

If we compare this score to the alignment score for all of the other unique sequences we could produce using *seq1* and *seq2*, shown in the table below, we see that this is the highest possible alignment score with *target*:

Possible sequence	Alignment score
1, 2, 7, 1, 3, 6	72
1, 2, 1, 7, 3, 6	84
1, 2, 1, 3, 7, 6	**100**
1, 2, 1, 3, 6, 7	99
1, 1, 2, 7, 3, 6	83
1, 1, 2, 3, 7, 6	99
1, 1, 2, 3, 6, 7	98
1, 1, 3, 2, 7, 6	97
1, 1, 3, 2, 6, 7	96
1, 1, 3, 6, 2, 7	80
1, 3, 1, 2, 7, 6	99
1, 3, 1, 2, 6, 7	98
1, 3, 1, 6, 2, 7	82
1, 3, 6, 1, 2, 7	72

Note: there could be more than one sequence that has the maximum alignment score.

As an example of one of the calculations in the table above, the alignment score of the sequence in the first row with *target* ((1, 2, 7, 1, 3, 6) with (1, 2, 1, 3, 7, 6)) is:

$$\begin{aligned}\langle(1,2,7,1,3,6),(1,2,1,3,7,6)\rangle &= 1(1)+2(2)+7(1)+1(3)+3(7)+6(6)\\ &= 1+4+7+3+21+36\\ &= 72\end{aligned}$$

Normalization

Cosine similarity should be a value between -1 and 1 indicating how closely aligned two sequences are, but when the sequences are not normalized, this score can be much larger than 1 or smaller than -1. A normalized version of this problem might look like *seq1* = (0.1, 0.2, 0.7), *seq2* = (0.1, 0.3, 0.6), and *target* = (0.1, 0.2, 0.1, 0.3, 0.7, 0.6).

Note that the cosine similarity of *target* with itself is 0.1(0.1) + 0.2(0.2) + 0.1(0.1) + 0.3(0.3) + 0.7(0.7) + 0.6(0.6) = 0.01 + 0.04 + 0.01 + 0.09 + 0.49 + 0.36 = 1, and this is also the cosine similarity of the solution with *target*. Meanwhile, the cosine similarity of *target* with the incorrect answer (0.1, 0.2, 0.7, 0.1, 0.3, 0.6) is 0.1(0.1) + 0.2(0.2) + 0.7(0.1) + 0.1(0.3) + 0.3(0.7) + 0.6(0.6) = 0.01 + 0.04 + 0.07 + 0.03 + 0.21 + 0.36 = 0.72.

You may notice that scaling the input vectors by a constant factor also scales the dot product. Because of this property and the fact that all possible answers are scaled by the same factor, we get the same sequence for the original problem above as we do for the scaled problem. Thus, even though the alignment scores we calculated in the unscaled problem are not technically cosine similarity, which must always be between -1 and 1, we still identify the same combination of *seq1* and *seq2* to solve the problem.

All of the test problems you will be given for this project will be normalized, and your final answers should be real numbers in the range -1 to 1 (up to round-off errors). That said, the subproblems you evaluate recursively will probably not have this property, and you **should not** try to normalize the inputs to the problem in your algorithm.

4 Project report

In your project report, you should include brief answers to 7 questions. Note that you must use dynamic programming to solve this problem; other solutions will not receive substantial credit.

1. How would you break down the problem of calculating the maximum alignment score for a combination of two sequences *seq1* and *seq2* (length n and m, respectively) with the sequence *target* (length $n+m$) into one or more smaller instances of this problem? Your answer should include what subproblems you are evaluating, as well as how you are using the answers to these subproblems to solve the original problem.
2. What are the base cases of this recurrence?
3. What data structure would you use to recognize repeated problems? You should describe both the abstract data structure as well as its implementation.
4. Give pseudocode for a memoized dynamic programming algorithm to find the maximum alignment score when combining *seq1* and *seq2* to align with *target*.
5. Give pseudocode for an iterative algorithm to find the maximum alignment score when combining *seq1* and *seq2* to align with *target*. This algorithm does **not** need to have a reduced space complexity relative to the memoized solution.

6. Can the space complexity of the iterative algorithm be improved relative to the memoized algorithm? Justify your answer.

7. Give pseudocode for an algorithm that identifies the sequence which will achieve the maximum alignment score. If there is more than one sequence that yields the maximum score, you may return any such sequence. Your algorithm may be iterative or recursive.

5 Coding your solutions

In addition to the report, you should implement a dynamic programming algorithm that can compute a maximum cosine similarity for a combination of two subsequences with a target. Your code may be iterative or recursive, but it must be a dynamic programming algorithm. You may code your solution in C++ or Java, but it must compile and run in a Linux environment. If you are using C++ and your code cannot be compiled using the command

```
g++ -o align *.cpp
```

you should include a Makefile that is capable of compiling the code via the `make` command.

If you choose to implement your code in Java, you should submit an executable jar file with your source. In either case, your source code may be split into any number of files.

Your code will not need to handle invalid input (e.g., non-normalized sequences or problems where the length of *target* doesn't equal the length of *seq1* and *seq2* combined).

5.1 Input format

Your program should read its input from the file `input.txt`, in the following format. The first line of the file has two positive integers n and m specifying the lengths of *seq1* and *seq2*, respectively. The second line will list the n real numbers (in the range -1 to 1) that make up *seq1*, the third line will have the m real numbers that make up *seq2*, and the fourth line will have the $n + m$ real numbers that make up *target*. All numbers on a line will be separated by spaces.

5.2 Output

Your program should write its output to the file `output.txt`. The first line of your output should contain the maximum cosine similarity for *seq1*, *seq2*, and *target*. The second line of your output should list the solution sequence $(n + m)$ that achieves this maximum cosine similarity, with consecutive values separated by spaces. (*seq1* and *seq2* should be subsequences of this solution sequence.)

6 Submission

Your submission for this project will be in two parts, the group submission and your individual peer evaluations.

6.1 Group submission

The submission for your group should be a zip archive containing 1) your report (described in Section 4) as a PDF document, and 2) your code (described in Section 5). If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. You should submit this zip archive to the "Project 2 (group)" assignment on Canvas.

Be aware that your project report and code will be checked for plagiarism.

6.2 Teamwork evaluation

The second part of your project grade will be determined by a peer evaluation. Your peer evaluation should be a text file that includes 1) the names of all of your teammates (including yourself), 2) the team member responsibilities, 3) whether or not your teammates were cooperative, 4) a numeric rating indicating the proportional amount of effort each of you put into the project, and 5) other issues we should be aware of when evaluating your and your teammates' relative contribution. The numeric ratings must be integers that sum to 30.

It's important that you be honest in your evaluation of your peers. In addition to letting your team members whether they do (or do not) need to work on their teamwork and communication skills, we will also evaluate your group submission in light of your team evaluations. For example, a team in which one member refused to contribute would be assessed differently than a team with three functioning members.

You should submit your peer evaluation to the "Project 2 (individual)" assignment on Canvas.

7 Grading

Report	**40 points**
Question 1, 4, and 5	8 points
Questions 2, 3, 6, and 7	4 each
Code	**30 points**
Compiles	5
Uses correct input and output format	5
Computes correct alignment score	10
Computes correct sequence	5
Good coding style	5
Teamwork	**30 points**

Note that if your algorithm is inefficient, you may lose points for both your pseudocode and your submission. Also, in extreme cases, the teamwork portion of your grade may become negative or greater than 30. In particular, if you do not contribute to your group's solution at all, you can expect to receive an overall grade of 0 on the project.