Project Requirements

The project accounts for 10% of the total grade of this course and is divided into three parts as below.

Implementation

Choose one of the algorithms below and implement it as detailed. You can use any programming language you want as long as you can satisfy all the requirements.

Linear-time selection algorithm

Implement the linear-time selection algorithm with median of fives pivot selection. The implementation should contain a function with the following signature.

float Select(float[] A, int n, int k)

A: is an array of floating-point numbers (both single- and double-precision are OK)
n: is the size of the array
k: is the rank of the element to find

The function should return the value of the kth element in the Array A.

You are allowed to use recursion. You don't need to worry about the case of wrong input, e.g., k > n.

Strassen Matrix Multiplication

Implement the Strassen Matrix Multiplication algorithm for multiplying two square matrices. It should not assume that n is a power of two. The function should have the following signature:

float StrassenMultiply(float[][] A, float[][] B, int n)

A, B: Are the two input matrices. Each input is a two-dimensional array of floating-point numbers with dimensions $n \times n$.
n: is the dimension of both arrays.

It should return the final product of the two matrices.

Matrix Chain Multiplication

Given the dimensions of a list of matrices, this function should compute the cost of matrix multiplication in terms of number of scalar multiplication operations and the final order of multiplications. Your implementation should include a function with the following signature.

int MatrixChainMultiplication(int[] p, int n)

p: is the list of matrices dimensions as described in class. The size of the array is always n+1.
n: is the number of matrices to multiply

The function should return the total cost in terms of the number of scalar multiplications. The function should also print the order of multiplications to standard output in the following format (as an example).

(((((A1)x(A2))x(A3))x(A4))x(A5))x(A6)

Note: You can use the lower-case x letter as a multiplication symbol for simplicity.

You should implement this function as a bottom-up iterative function. You are not allowed to use recursion.

Edit Distance

Given two strings, compute the edit distance, i.e., the minimum number of operations to transform the first string to the second one. It should also print out the sequence of operations that transforms x into y. The function should have the following signature.

int EditDistance(String x, String y)

x, y: The two input strings

The function should return the minimum edit distance between the two strings. The function should also print out the sequence of operations that transform x into y. For example, for the two inputs "clarks" and "kirk", it should print out something like the following.

Substitute "c" into "k"
Delete "l"
Substitute "a" into "i"
Delete "s"

Dijkstra's Algorithm

Given a weighted graph G=(V,E), a weighting function W, and a starting vertex s, find the single-source all-destinations shortest path. You can assume that all weights are non-negative. The function should have the following signature.

DijkstraShortestPath(Map<Int, Int[]> E, Map<(Int, Int), Float> W, int[] p, float[] d);

E: Is the list of edges in adjacency list representation.
W: Is the weighting function as a map from an edge to its weight value.
p: Is the output array that will contain the shortest path tree topography (predecessor).
d: Is the output array that will contain the shortest path distance to each vertex.

Bellman-Ford Algorithm

Given a weighted graph G=(V,E), a weighting function W, and a starting vertex s, find the single-source all-destinations shortest path. Some weights might be negative. The function should have the following signature.

Boolean BellmanFordShortestPath(Map<Int, Int[]> E, Map<(Int, Int), Float> W, int[] p, float[] d);

E: Is the list of edges in adjacency list representation.
W: Is the weighting function as a map from an edge to its weight value.
p: Is the output array that will contain the shortest path tree topography (predecessor).
d: Is the output array that will contain the shortest path distance to each vertex.
The return value is true if no negative cycles were detected. The function should return false if it detects any negative-weight cycles. In the latter case, the value of the two output arrays is undefined.

Presentation

You should prepare a two-minute video to present your implementation and mention at least one lesson you learned from implementing the algorithm or a piece of advice you have for others implementing this algorithm. The video should contain the following. The timing shown below is a suggestion and you can adapt based on your actual content.

- (10 seconds) Introduce yourself. There should be a slide clearly stating which algorithm you implemented as a title, your name, and the course number and name.
- (20 seconds) Introduction to the problem. Since we are all familiar with all the above problems, present it in no more than 20 seconds.
- (60 seconds) Present the main part of your implementation. You do not have to go through the entire implementation but choose a part that you believe is interesting and describe it. You can use this part to describe the lesson you learned.
- (30 seconds) Conclude by showing a screen recording of your code running on an example. Show the output of the program. You can use one of the examples we had in class or create your own example. Do not use a trivial example that does not fully run your algorithm.

Q&A

The last part of the project is to watch each other's presentation and answer the questions. All the presentations will be compiled into a single video with a total time of about three hours. You will need to go through the video and answer one question for each presentation. If you answer 80% of the questions right, you get full mark. You will have the flexibility to start the video at the time that suits you within a give time window. However, you will be allowed one submission only so make it count.

Deliverables

You will need to deliver the following.

- The source code of the program you wrote. Make sure that the code is readable and well-documented.
- A README file that includes your name, student ID, and simple instructions to explain how to compile and run the code.
- A two-minute recorded video. Any video longer than two minutes will be trimmed to two minutes and you will lose points.
- A single question, three choices, and the answer to your question. Notice that the answer to your question should be available in the presentation and should not require doing extra work. The presentation should not explicitly mention the question and/or the answer. The presentation should be natural and the answer should be part of the flow.

Submission Instructions

- Include the source code and the README file in a single compressed file in either .tar.gz or .zip formats. Other formats including .7z and .rar are not accepted and may result in losing points. Name the file as “<UCRNetID>-Project-Code” and replace <UCRNetID> with your UCR NetID, e.g., bond007. Do not include the angle brackets or double quotes in the file name.
- The video should named “<UCRNetID>-Project-Video” with extension .mov or .mp4. Any other format is not accepted and may result in losing some points. A video longer than two minutes will be trimmed to two minutes and will result in losing points.
- The question should be provided as a **plain text file** named “<UCRNetID>-Project-Question.txt”. Any other format, e.g., .docx or .pdf, will not be accepted and may result in losing points. The contents of the file should be **exactly four lines with** no empty lines. The first line is the question, and the three following lines are the three choices. The first choice should always be the correct one. The choices will be shuffled when added in the final exam. If you want to include an equation in math form, write it in Latex style between “\(” and “\)” with the backslash and without the double quotes. Do NOT use any numbering of the choices, e.g., 1, 2, 3 or a, b, c.

Rubric

(12 points) Project Code

- (4 points) Correctness
- (4 points) Readability and style
- (4 points) Following the method signature given above and the naming conventions

(8 points) Presentation

- (3 points) Following the given time and the naming conventions
- (3 points) Presentation contents and lesson(s) learned
- (2 points) Correctness and clarity of the given question

(20 points) Q&A

- Formula: Min(20, 25*(# of correct answers)/(# of questions))

Due Date

The source code, presentation, and question are all due by Monday 6/12 at 11:59 PM. The combined video and questions will be released on Wednesday, 6/14, and will be due by Friday 6/16 5:00 PM.