CSE 6140 / CX 4140 Assignment 3
due Oct 11, 2018 at 11:59pm on Canvas

Please upload a single PDF named `assignment.pdf` for all of your answers/report containing:

1. a typed preamble that contains:
   (a) the list of people you worked with people for each question (if applicable),
   (b) the sources you used,
   (c) and if you wish, your impressions about the assignment (what was fun, what was difficult, why...);
2. your solutions for all problems, typed (not handwritten)

1 Divide and Conquer

Let A be an array of n relative integers. We want to find the maximum sum of contiguous elements, namely, two indices i and j $(1 \leq i \leq j \leq n)$ that maximize $\sum_{k=i}^{j} A[k]$.

a) If the values in the array are $A[1] = 2, A[2] = 18, A[3] = -22, A[4] = 20, A[5] = 8, A[6] = -6, A[7] = 10, A[8] = -24, A[9] = 13$, and $A[10] = 3$, can you return the two indices and the corresponding optimal sum?

b) Design an algorithm that returns the maximum sum of contiguous elements with a divide-and-conquer algorithm.

2 Master Theorem

For each of the following recurrences, give the asymptotic solution if the recurrence can be solved with Master Theorem. Otherwise, briefly explain why Master Theorem is not applicable.

a) $T(n) = 49T(\frac{n}{7}) + n^2 + 3n$

b) $T(n) = \frac{1}{4}T(\frac{n}{9}) + n$

c) $T(n) = 4T(\frac{n}{2}) - n^2 - 2n$

d) $T(n) = 2T(\frac{n}{4}) + 1000n^{0.6}$

e) $T(n) = 3T(\frac{n}{2}) + \ln 2$

3 Dynamic Programming: George Learns Algorithms

George P. Burdell has been taking the algorithms course at Tech every year since the College of Computing opened in 1964. As he looks at his transcripts over the decades, he wonders if he has been getting the hang of the material. The difficulty of the course varied based on which instructor taught it and how much time George spent at Georgia Tech football games, so naturally his final scores fluctuated from year to year. He decides the best way to measure his progress is to look at the sequence of final scores he earned (thankfully they are all positive and $\leq$100) and find the longest subsequence of scores over which his grade strictly improved. He looks at his grades from 1980-1989:

$$\{82, 77, 65, 89, 83, 68, 88, 71, 91, 90\}$$

George does not want to restrict himself to only contiguous years, since then it looks like he is never able to improve his performance for longer than 2 years ($\{65, 89\}$ or $\{68, 88\}$ or $\{71, 91\}$). If he allows himself to leave out whatever years he likes, he could see his grade improving over the following 3 scores: $\{77, 89, 91\}$. Or better yet, the following 4 scores: $\{77, 83, 88, 91\}$.

Feeling better about himself already, George defines a subsequence to be the original sequence with some (or all) of the elements removed. Now, he enlists the help of one his clever classmates from this year's algorithms class (you) to find his longest subsequence of improving grades.

1. George is not convinced that his problem can be solved any way besides a brute force search through all 2^n possible subsequences (where n is the number of times George took algorithms). Prove this problem has optimal substructure to show George there may be a better approach.

2. Write down a recurrence relation for this problem.

3. Design a bottom up DP algorithm to solve the problem. Include the time and space complexity of this algorithm to conclusively show George that your algorithm is better than an exhaustive search.

4 Optimizing Deep Learning

Deep learning is an emerging field in Computer Science that is finding applications with a wide spectrum of tasks, from identifying cats in images to beating Lee Sedol at Go. Almost all deep learning architectures rely on multiplying a large number of huge matrices.

Let us assume that the time required to multiply two matrices, with dimensions (p, q) and (q, r), is simply the number of scalar multiplications pqr. You may also remember that matrix multiplication is associative, i.e., for matrices A, B, and C, $(AB)C = A(BC)$. As such, the time to compute the result of multiplying a series of matrices will depend on the order of multiplication.

For example, if the dimension of matrix A is $(10, 20)$, matrix B is $(20, 5)$, and matrix C is $(5, 30)$, then the time for computing $(AB)C$ will be $(10 \times 20 \times 5) + (10 \times 5 \times 30) = 2500$ while that for computing $A(BC)$ will be $(20 \times 5 \times 30) + (10 \times 20 \times 30) = 9000$.

More generally, given a series of matrices to multiply, one can enumerate all possible parenthesizations of the series in order to determine the multiplication order which takes the least total time. Can a more optimal algorithm be designed for the purpose? Justify your answer.