CS218 Design and analysis of algorithms **Jan-Apr 2024**

Programming Assignment 2

Total Marks: 5

Most computational problems you would encounter in real world will be NP-hard, and thus, unlikely to have a polynomial time algorithm. This programming assignment may be one such example (not completely sure). For NP-hard problems, people still try to design efficient exponential time algorithms which run fast on realistic inputs. That is the goal of this assignment.

There is a simple algorithm for the problem given below, that runs in time $O(n! \times n^3)$. This will be too slow on some of the given inputs. To solve all the given inputs, you should aim for a running time of $O(2^n \times \text{poly}(n))$. Some dynamic programming ideas may be helpful.

You can ignore the last input. It is given just in case someone manages to get a polynomial time algorithm. You are highly encouraged to think about a polynomial time algorithm. It may get you a million dollars. :)

Scheduling processes with minimum peak memory requirement. We are given a set of n processes, some of which are dependent on others. These dependencies can be represented by a directed acyclic graph. Each process requires some input data from the processes on which it is dependent, and generates some output data. We are given the amount of data $m(i, j)$, which process j needs from process i. If j does not depend on i then $m(i, j) = 0$. We are also given the memory requirement to run each process, say $m(i)$ for process i. We can schedule these processes in any order that is consistent with the given dependencies. That is, any topological ordering of the underlying directed graph is a valid schedule. If we schedule the processes in order $(i_1, i_2, \ldots, i_n)$, then the peak memory requirement is

$$\max_{k \in \{1,2,\ldots,n\}} \left\{ m(i_k) + \sum_{1 \le p < k} m(i_p, i_k) + \sum_{k < q \le n} m(i_k, i_q) + \sum_{1 \le p < k} \sum_{k < q \le n} m(i_p, i_q) \right\}.$$

We want to find a valid schedule that minimizes the peak memory requirement. Assume that $m(i, j) = 0$ for $i > j$, which means $(1, 2, \ldots, n)$ is a valid schedule.

Instructions

Input contains $n + 1$ lines.

Line 1: n (the number of processes)
Line 2: $m(1)$ $m(1,2)$ $m(1,3)$ $\ldots$ $m(1,n)$
Line 3: 0 $m(2)$ $m(2,3)$ $\ldots$ $m(2,n)$
Line 4: 0 0 $m(3)$ $m(3,4)$ $\ldots$ $m(3,n)$
$\vdots$
Line $n+1$: 0 0 $\ldots$ 0 $m(n)$

Output : the minimum possible value of the peak memory requirement

- Programming Language: C++. We will compile your code with g++. Make sure that it works.
- Submission: put your code in a file named XXX.cpp where XXX is your roll number. Also, write a short explanation (a paragraph) of what your algorithm does, put this in XXX.pdf. The two files should be uploaded on Moodle (do not zip/compress).
- Given files: In the `MinPeakMemory` folder, you will find: (i) helper.cpp (a c++ code showing expected input/output, feel free to use) (ii) Few sample input and output files, (iii) two executable files, which can be used to get the correct output on any input.

- Running time: we will test your code on some similar size instances as given in the sample input files (few of small size, few of large size). To solve all the given inputs, you should aim for a running time of $O(2^n \times \text{poly}(n))$.
- Academic integrity: Mention all references if you have referred to any resources while working on this assignment in the pdf. You are supposed to do the assignment on your own and not discuss with anyone else. We will do a plagiarism check on your submission using MOSS. It's fairly sophisticated and can detect even when you have made modifications in someone else's code. Any cases found with significant overlap will be sent to DADAC. If DADAC finds it to be a case of plagiarism, then the penalty is zero in the assignment and final course grade reduced by 1 point.
- Grading: We will use mars.cse for testing, with a timeout of 5 seconds for each input. The test inputs will be of varying sizes. Total marks will be equally distributed for the test inputs.