J. Ghosh
Due by: Wed May 13, 2015, 11:59PM Austin Time.
Total Points: 40

Programming assignments are to be done individually. You may discuss the problem and general concepts with other students, but there should be no sharing of code. You may not submit code other than that which you write yourself or is provided with the assignment. This restriction specifically prohibits downloading code from the Internet. If any code you submit is in violation of this policy, you will receive no credit for the entire assignment. Submissions received after May 13, 2015 at 11:59pm will receive no credit for this assignment.

This project is about running complex software programs that interact with the environment in (near)-real time situations. Processes may have dependencies among themselves. Moreover, as different processes complete execution they may enable certain "actuators", and there may be desired times by which such enablement should happen, otherwise penalties may be incurred. Also, if there are a limited number of processors available to run the processes, additional delays can be incurred if there is no available processor to run a "ready" process. The problem space is actually quite large, and we shall explore only certain aspects of it.

There are three parts to this project, with the number of processors available being one, very high and a small value greater than one, respectively. For all three parts, assume that processes cannot be pre-empted, i.e. once a process starts running, it has to complete execution before one can begin executing another process on the same processor. Also processes are atomic in that they cannot be further divided into smaller pieces for concurrent execution.

1. **(2+8+2=12 pts total)** A complex software program consists of a set of N processes, that will run (or "execute") on a single processor. Once begun, each process is executed without any interruptions till it is completed. The i^{th} process takes time $t_i > 0$ to execute, $1 \leq i \leq N$.

   Let us assume for this question that the processes have no dependencies at all and thus can be executed in any order. However, for each process i there is a desired time d_i such that it is considered to be late by an amount $C_i - d_i$ if the completion time C_i is more than d_i. The completion time of a process depends on the order in which the processes are run, which is your design choice. The goal is to find an order of executing the processes so as to minimize the total "lateness", i.e. find the minimum value of $\sum_{i=1}^{N} max(0, C_i - d_i)$. Note that there are $N!$ distinct ways of ordering the processes. It turns out that this problem is not easy, even a dynamic programming (DP) approach has an exponential complexity (though better than N!).
   (i) First, show using a counter-example, that the greedy solution of ordering the processes based on deadline ("earliest deadline first", which works for the "minimizing maximum lateness problem") is not guaranteed to be optimal for this problem.

   (ii) Briefly describe the logic behind your algorithm (in English) to solve the "minimizing total lateness" problem. Then write a psuedo-code for such a DP, followed by actual code that you will run using the data in the input file. A sample "test" input file is included in the project folder to let you know what format the input will be provided in. You should also provide the answer

that you obtain for the test input file provided to you.
(ii) What is the asymptotic time complexity of your algorithm?

2. **(2+3+2+10+3=20 pts total)** Now suppose you want to run (or "execute") your program containing N processes, on a multicore machine with M identical cores. Each process is executed on a single core, and without any interruptions till it is completed. As before, the i^{th} process takes time $t_i > 0$ to execute, $1 \leq i \leq N$. However, some of these processes are now dependent on the outputs of other processes, and cannot be started till all of the processes they depend on are completed. For each process, ki, this information is provided in a dependency list D_i, which is simply a list of all process i.d.'s on which the i^{th} process depends on. Note that D_i may contain a value $j > i$, i.e. the i.d. numbers may not directly reflect the dependencies.

   Suppose $M = N$, so there is always a core available for any process that is ready for execution. Hence a schedule that minimizes T_N, the total time taken to execute the entire program, is trivial: a process starts being executed as soon as all the processes whom it is dependent on complete. However, you also want to determine the smallest value of T_N achievable, and this will require designing an algorithm A. ***Hint-*** *Construct a precedence graph using the given dependencies, and consider the relationship between the given problem and this graph.*

   (a) Briefly describe your formulation of the solution (Algorithm A) in English.

   (b) Write pseudocode of your solution. The inputs to the algorithm will be the t_i's and the D_i's.

   (c) What is the asymptotic time complexity of your algorithm ?

   (d) Write a program in Java or C++ to implement your algorithm. Provide the answer for the test input data available on Canvas and include it in your report. Go through the *Instructions* for the input data format and the expected output of your code. In both your pseudocode and actual code, you should include a check to see that there is a feasible solution for any given input, i.e. whether the input is valid or not, before you attempt to find the minimum time.

   (e) Prove the correctness of your algorithm.

3. **(8 pts)**

   The assumption $M \geq N$ in question 2 seems unrealistic. Instead suppose that there are only 3 cores. It turns out that this makes the problem intractable!

   Provide a "practical" (i.e., a good heuristic) algorithm (English description/rationale, followed by pseudocode and source code) to schedule processes to processors for $M = 3$. Let T_3 be the time taken by the schedule obtained by your scheme for a specific set of processes. Let T_{3B} the corresponding time taken by a "baseline" solution for the 3-core case, where one simply schedules according to process i.d., i.e., when a processor gets free it starts running the process with the smallest index i which has not been executed yet. Compare T_3, T_{3B} and T_N (from Q.2) for the input file provided through Canvas. You will be graded on the quality, efficiency and innovativeness of your solution.

Instructions

- This programming assignment is to be done individually. There should be no sharing of code.
- You can code in C/C++ or Java. You can use the standard library of whatever language you code in.
- Submit all the files - source code, report as a single zip file on Canvas. Don't forget to put your name, in fact it is helpful if your name is also reflected in the filename (e.g. Lastname.pdf) itself.
- Your source code should read the input data from a text file. Your program will be called with the file name (that includes the full path) as its argument.
- Your output should be as follows.
  - **Problem 1:** Your code should output a list of process IDs sorted by their scheduled start times. Report the obtained scheduled start times for each process. Also report the value of $\sum_{i=1}^{N} max(0, C_i - d_i)$ of the schedule obtained by your scheme.
  - **Problem 2:** Your code should output a list of process IDs sorted by their scheduled start times. Report the obtained scheduled start times for each process. Also report the value of T_N taken by the schedule obtained by your scheme.
  - **Problem 3:** Your code should output a list of process IDs sorted by their scheduled start times. Report the obtained scheduled start times for each process. Also report the time T_{3B} taken by the schedule obtained by your scheme.
- For question 3, you do not need to output the processor ID to which a particular process gets assigned.

Input data

The format of the input files are as follows.

- **Problem 1:** See `p1-testcase.txt` as a sample input file. The first line of the input file is the number of processes (N), the N next lines each has the following format:

  processID execution time deadline

  These values are tab-separated.
- **Problem 2:** See `p2-testcase.txt` as a sample input file. The first line of the input file is the number of processes (N), the N next lines each has the following format:

  processID execution time dependency list

  These values are tab-separated.

- **Problem 3:** See `p3-testcase.txt` as a sample input file. The first line of the input file is the number of processes (N), the N next lines each has the following format:

  processID execution time dependency list

  These values are tab-separated.

- Process ID is an integer between 0 and N-1.
- Execution time and deadline are integers.
- List of dependencies is comma separated and is inside {}