CmpE300 Analysis of Algorithms

Project 2

Instructor: Tunga Güngör, gungort@boun.edu.tr

TA: Kutay Altıntaş, kutay.altintas@boun.edu.tr

Announcement Date: 6/12/2023

Due: **24/12/2023 23:59 Sharp**

1 Introduction

Industry 4.0, often referred to as the Fourth Industrial Revolution, marks a significant transformation in the industrial sector driven by technological advancements. It integrates modern information and communication technologies to create smart, automated production environments. The core elements of Industry 4.0 include the Internet of Things (IoT), cloud computing, artificial intelligence, and big data analytics. These technologies facilitate highly efficient, flexible, and customizable manufacturing processes, enabling real-time data exchange and automation in factories.

A Digital Twin is a virtual model of a physical system, process, or product. It's a cornerstone concept in Industry 4.0, representing a dynamic digital replica of physical assets. Digital twins use data from sensors embedded in real-world objects to simulate their behavior in a virtual environment. This technology allows for real-time monitoring, predictive maintenance, and in-depth analysis of systems. Implementing a digital twin in a factory setting involves creating a comprehensive virtual representation of the manufacturing processes, enabling the simulation and optimization of those processes, and enhancing decision-making.

In this project, you are a factory manager of a hypothetical company. In order to achieve your company's Industry 4.0 goals, you decided to create a digital twin of your factory. Your factory consists of flexible machines that can perform different operations on different products. This flexibility makes it difficult to predict which products are produced given initial settings. In addition to this, maintenance costs are unpredictable due to the flexibility of the machines. The digital twin of your factory must be able to simulate your production processes and keep track of maintenance operations. This will be a good starting step for the next industrial revolution.

The goal in this project is to create a parallel algorithm to simulate a factory. Machines can run parallel in a factory and communicate with each other. The properties of the machines, the factory, and the extend of communication is explained in the following sections.

2 Digital Twin

A factory runs many processes parallel to each other. To simulate the factory better, and to increase efficiency of the simulation, you must implement a parallel algorithm using MPI. The factory consists of various machines connected to each other, and each machine needs to be implemented as a worker/slave process. These processes must communicate with each other.

Your factory has a main control room. The control room decides the initial settings of the machines and production parameters. The control room also keeps track of the maintenance costs of the machines. You must create a master process to represent the control room.

2.1 Products

Products are the things machines are processing. A machine receives some products, performs some operations on them, and passes its output product to another machine. Eventually, your factory produces a final product. To clarify the terminology, suppose that your factory produces cars. In this case, car is the final product. The products in the factory can be engines, seats, glass panes etc. In this project, products and final products are pieces of strings.

Initially, some of your machines must begin production with some products. These machines are explained in the Machines section. You may assume that these machines are being fed products by the factory workers, and this process is not a part of your simulation. Products are pieces of strings consisting of capital letters from A to Z. Example products are "AFKEC", "CE", "A", "JECELACKOOO". The machines process these products and create new products. Eventually, the terminal machine (the last machine in the production layout) creates a final product using received products. A final product is no different from products, except that it will not be further processed by any machines. Please pay attention to distinction between products and final products in your report to avoid any confusion.

2.2 Machines

The layout of your factory is a tree structure. A machine may receive multiple products from other machines, but it passes its output product to a single machine. Finally, products arrive at a terminal machine (root node of the tree). This machine produces the final product using received products, and informs the control room about which final product is produced. Machines have unique id numbers assigned to them, and they perform different operations in different production cycles.

2.2.1 Operations

The machines perform various operations. These operations are *add*, *enhance*, *reverse*, *chop*, *trim* and *split*.

- **Add:** The machine concatenates received products. The concatenation must be done in the order of the child machines id numbers. Example: Machine 7 receives product "ABC" from machine 10, "DEFG" from machine 12, and "ZZZ" from machine 11. If machine 7 performs *add* operation, Then it must produce "ABCZZZDEFG" ("ABC"-10 + "ZZZ"-11 + "DEFG"-12).
- **Enhance:** The machine duplicates the first and the last letter in the product. Example: "ABC" → "AABCC", "YU" → "YYUU", "A" → "AAA".
- **Reverse:** The machine reverses the product string. Example: "ABCD" → "DCBA".
- **Chop:** The machine removes the last letter from the product. If the product consists of a single letter, then this operation is not performed, but the machine still wears out (explained in the Maintenance section). Example: "ABCD" → "ABC", "A" → "A".
- **Trim:** The machine removes the first and the last letters from the product. If the product consist of two or one letters, then this operation is not performed, but the machine still wears out (explained in the Maintenance section). Example: "ABCD" → "BC", "ABC" → "B", "AB" → "AB".
- **Split:** The machine splits the product into two parts, and discards the right part. If the product consists of odd number of letters, the split operation is performed after middle letter. Example: "ABCD" → "AB", "ABCDEFG" → "ABCD", "A" → "A".

All machines perform *add* operation when they receive a product. Then any of the other operation is applied according to rules specified below. The terminal machine performs only the *add* operation and performs no other operations. As mentioned previously, each machine has a unique id assigned to it. If a machine id is an odd number, then this machine performs only *reverse* or *trim* operations in a production cycle and alternates between them for each cycle. This means that if a machine performs *reverse*, then it performs *trim* for the next received products, and then *reverse* and so on. Please note that regardless of the machine id and previous operations, all machines perform *add* operation initially. For machines with even number id's, they alternate between *enhance*, *split*, and *chop* in this order.

The initial operation to be performed for each machine is set by the control room. After they are being set, they alternate between allowed processes. If machine 7 is set to *trim*, then it performs *trim* → *reverse* → *trim* → *reverse* ... For instance If machine 10 is set to *split*, then it performs *split* → *chop* → *enhance* → *split* → *chop* ...

2.2.2 Communication

Machines communicate with each other in a blocking way. If a machine has four children, and receives only two products from two machines, then it must wait for the other two children to begin its operations. You must use your MPI library's blocking communication functions such as *Send()* and *Recv()*. Lastly, your terminal machine must inform the control room about the final product.

2.3 Maintenance

Machines wear out as they perform their operations. After a certain threshold, they must notify the control room that they need maintenance. You may assume that maintenance happens instantaneously and machines perform their operations as usual. The simulation is not intended to simulate breakdown times. It aims to predict costs of maintenances, which depends on various factors.

Initially, you will be given an integer value *wear factor* for each operation and a *threshold*. *Wear factors* represent how much a machine wears out after performing that operation. As machines perform their operations, the wearing accumulates. If the wearing reaches the threshold or exceeds it, then the machine needs maintenance. The machine must calculate its maintenance cost and send it to the main control room. The cost of maintenance is calculated according to this formula:

$$C = (A - T + 1) * WF, \quad A \geq T$$

In the formula, C denotes cost of the maintenance, A denotes accumulated wear, T denotes the threshold value, and WF denotes wear factor of the last performed operation. The machine subject to maintenance must send a string in the form of <Machine Id>-<C>-<Current Production Cycle> to the main control room using non-blocking communications. An example string is 4-9-2 (Machine 4 has maintenance cost 9 at production cycle 2). The main control room keeps track of these logs. After a machine receives maintenance, its accumulated wear is reset to 0.

Lastly, *add* operation does not have a wear factor. This means that the terminal machine will never wear out and will never need maintenance.

3 Initialization

You will be given an input text file that contains the necessary information. In this file, the number of machines, wear factors for each operation, maintenance threshold, initial operations of the machines, and the links between machines are provided. In addition to this, you will be given an integer that denotes the number of production cycles. Each machine must run exactly equal to the production cycle number. This means that in the end, you will produce number of products equal to the production cycle number. Be careful not to perform further operations, as it may inflate the maintenances and result in a wrong answer. Lastly, you will be given a list of products equal to the number of leaf machines (machines that no other machine precedes them). These machines will receive given products and begin production.

The main control room, which is the master process of your program, must read the input text file, and initialize everything. Worker/slave processes for each machine must be spawned by the master process, and the necessary information must be passed to them.

Here is an example input file for your program:

```
7
10
4 3 1 2 3
13
2 1 enhance
3 2 trim
4 2 chop
5 3 trim
6 3 split
7 4 trim
IE
OX
YX
```

The first line represents the number of machines. In this example, there are 7 machines. The second line represents number of production cycles. The third line denotes wear factor for each operation, in the order of *enhance* (4), *reverse* (3), *chop* (1), *trim* (2), *split* (3). The fourth line is the threshold value for maintenance. The subsequent 6 lines, the number of machines minus 1, is an adjacency list of machines. In each line, the first number is a child machine's id, and the next number is its parent machine's id. Since this is a tree structure, and each machine has only one parent except the terminal machine, the number of edges (lines) is equal to the number of machines minus 1. The initial operation for a child machine is given next (Machine 2 starts with *enhance*, machine 3 starts with *trim*...). Lastly, the subsequent lines are products received by leaf machines. In this example, machines with id's 5, 6, 7 do not have any other machine preceding them (meaning that they are leaf machines). Therefore, machine 5 processes product "IE", machine 6 processes product "OX" and machine 7 processes product "YX". These products must be assigned to leaf machines by the order of their id's, not in the order of their appearance in the adjacency list.

Figure 1 demonstrates the first production cycle for the example input. For instance, Machine 3 has Machine 5 and Machine 6 as its children, and has Machine 2 as its parent. Machine 3 receives "IE" from Machine 5 and "O" from machine 6. First it adds them and produces "IEO". Then it performs trim operation and produces "E" as its output. Then it passes "E" to Machine 2. In the next production cycle, Machine 3 must perform *reverse* operation after *add* operation. Lastly, terminal machine (Machine 1) produces "EEYY" as the final product, and sends it to main control room. The final product for the first cycle is "EEYY".

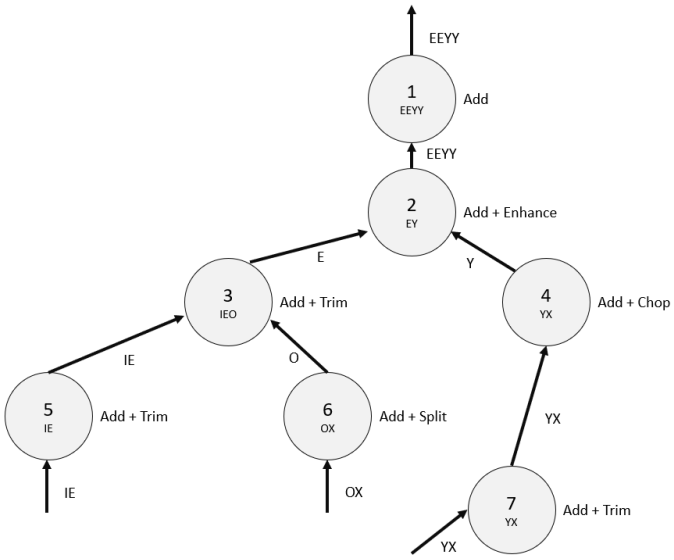

Figure 1: The first production cycle for the example input.

4 Output

The main control room must write produced products in correct order to a text file. After these products, the control room must write maintenance logs to the same file. The order of the logs may change.

Your code will be run with two arguments, input file location and output file location. As an example, assuming that you did your project in Python, the following command will be run to test your code:

```
mpiexec -n 1 python yourcode.py input.txt output.txt
```

Please note that your code will be tested with different test cases, and if you did your project in another programming language, see Deliverables section.

5 Deliverables

You must deliver your code files with a README file. You must write which command should be used to run your code, and list all libraries that need to be installed. If your code does not run by your command, then you will not receive any points for this part of the project.

Please use plenty of comments in your code. Use understandable names for your variables and functions. In each of your code files, write your names, students numbers and group number as a comment.

Write a report for your project. Do not use any LLM's for your report or code. In your report, explain your design choices. Describe the advantages of using parallel programming for this project. Create a mock input with no less than 7 machines, and show the first production cycle using diagrams. If your code does not work/compile, or does not give intended results, please write it in your code. Explain what might went wrong. Lastly, if this was a real Industry 4.0 project aiming to implement a digital twin, what would be the challenges for the implementation? The last question is a bonus question (+10 pts), please don't write more than half a page for it. You must submit your report as a .pdf file.

In addition to the project files, each student in a group must submit a .pdf document stating explicitly the parts of the project she/he has worked on. The document must begin with the line "I worked on the following parts in this project:" and all the parts the student has worked on must be explained very clearly and in not less than 10 lines. Do not write general comments such as "We worked on the project together." or similar comments. Write what you have done in the project explicitly. If both students write the same or similar explanations, their projects will not be graded.

In brief, you are expected to submit three files. You must submit a .zip file for your program (codes, README etc.). You must submit a report file as .pdf. Each group member must submit a separate document as .pdf explaining their contribution.