សាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ

ROYAL UNIVERSITY OF PHNOM PENH

Multi-threaded and Single-threaded Program

Lecturer: Kor Sokchea

Submitted by: Tang Lymeng

January 2023

TABLE OF CONTENTS

I. Introduction

The objective of this project is to develop a multi-threaded sorting program that divides a list of integers into two smaller lists of equal size and sorts each sub-list using a sorting algorithm of our choice. Then, a merging thread merges the two sorted sub-lists into a single sorted list. We implemented the program in Java, and compared its performance with a single-threaded version of the same program.

II. Objective

The main objective of this project is to explore the advantages of multi-threaded programming by developing a program that sorts a list of integers using multiple threads. We aim to measure the performance of the multi-threaded program and compare it with the single-threaded version of the program to determine if multi-threading improves the sorting speed.

III. Implementation

3.1 Multi-threaded sorting program

To implement the multi-threaded sorting program, we will first create a global array of integers and divide it into two sub lists of equal size. We will then create two separate threads, one for each sub list, and pass the starting index for each thread as a parameter. These sorting threads will then use the bubble sorting algorithm to sort the sub lists. Once the both threads done their works, we will create a third threaded and named it merging threaded, which will merge the two sub lists into single list.

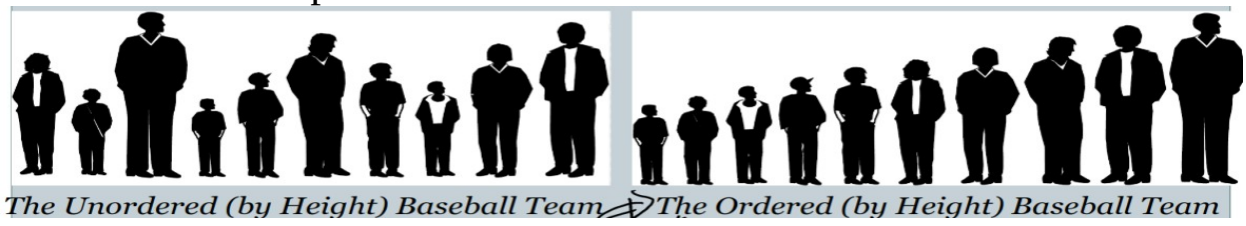
3.2 Single-threaded sorting program

For single-threaded sorting program, we will simply implement the bubble sorting algorithm in the main function without creating any threaded.

- Bubble sort algorithm

  Sorting is a technique that arrange names in alphabetical order, put elements of a list into an order, etc.

  - For example of unordered and ordered data



  - How Bubble sort work?

1. You start at the left end of the line and compare the two kids in positions 0 and 1.
2. If the kid on the left (in position 0) is taller, you swap them. If the kid on the right (in position 1) is taller, you don’t do anything.
3. Then you move over one position and compare the kids in positions 1 and 2.
Again, if the kid on the left is taller, you swap them.

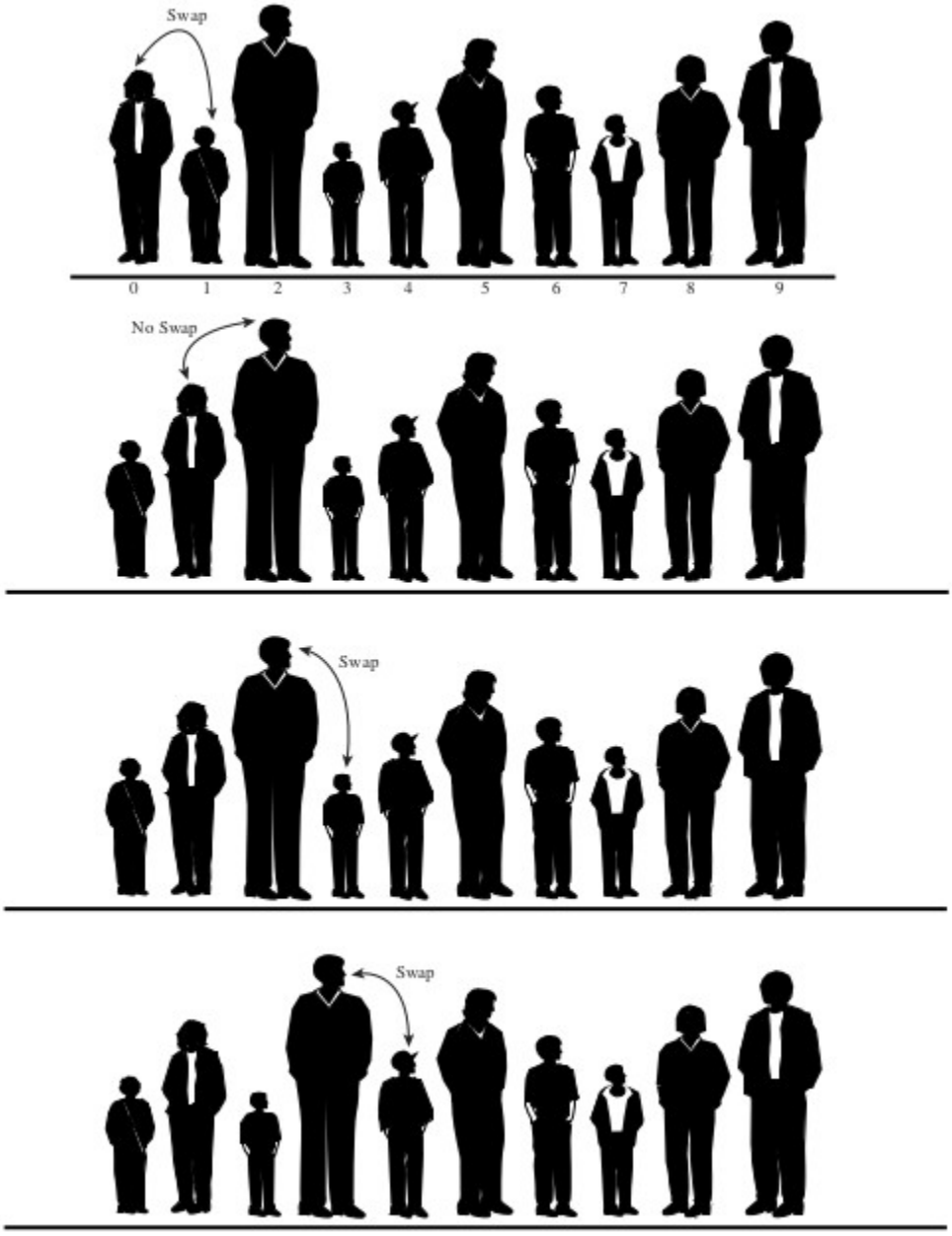


Figure 1: beginning of first pass

Here are the rules you’re following:
1. Compare two players.
2. If the one on the left is taller, swap them.

3. Move one position right.

Figure 2: end of first pass

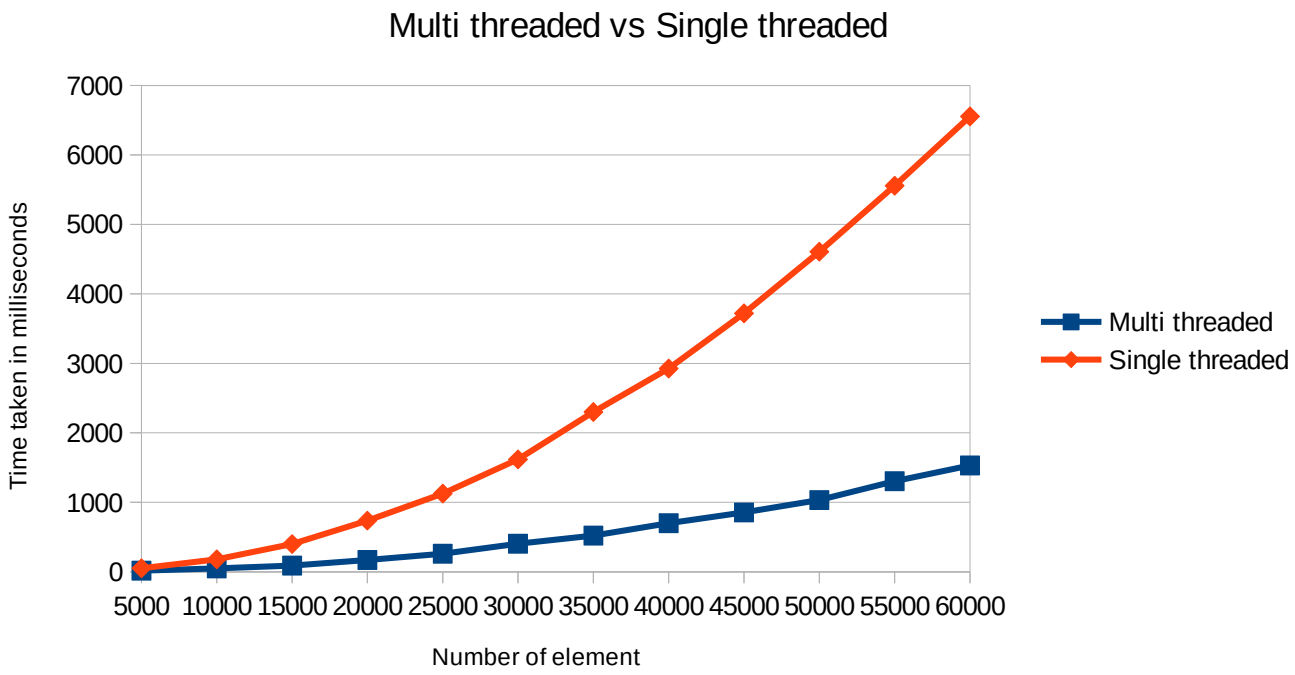
4. When you reach the first already-sorted player, start over at the left end of the line.

Continue this process until all the players are in order.

IV. Result

I have testing both of the program multi-threaded and single threaded with random number between 1-1000.

No. of element(random number 1-1000)	Multi threaded	Single threaded
5000	17	52
10000	49	180
15000	90	400
20000	170	737
25000	261	1126
30000	404	1618
35000	522	2300
40000	700	2926
45000	855	3721
50000	1033	4607
55000	1304	5556
60000	1531	6555



V. Conclusion

In conclusion, the multi-threaded sorting program we developed was successful in improving the sorting speed of a list of integers. The program was able to sort the list almost fourth time as fast as the single-threaded program. This demonstrates the advantages of multi-threading in improving program performance, especially for tasks that require parallel processing. Although Bubble Sort is not the most efficient sorting algorithm, it served our purpose of demonstrating the advantages of multi-threading. In future work, we could explore other sorting algorithms and measure their performance in a multi-threaded program.