

Name, SID, Date

In Class Assignment 26: Multi-threaded Merge Sort

Benjamin Sanders, EdD, MS April 5, 2024

1 Introduction

You will need to work individually to complete this assignment. Write your name at the top of all pages for this assignment. Turn in all work to Blackboard on or before the deadline to receive credit.

You may use additional libraries and online resources, if you get them approved in writing, over email, from the instructor first. If you have received approval from the instructor, write the approved libraries and any references in the space below.

.....
.....
.....
.....

2 Assignment Description

2.1 Big Picture

This exciting style of algorithm can utilize multiple cores in a processor for a significant performance improvement. Further, if you are ever interested in programming directly onto a graphics card, this parallel or multi-threaded style is how you should expect to program.

2.2 Algorithm Implementation

Implement the following algorithm in Java, using the Vector data structure for any 1-D array, 2-D array, or linear algebra purposes.

P-MERGE-SORT(A, p, r, B, s)

```
1   $n = r - p + 1$ 
2  if  $n == 1$ 
3       $B[s] = A[p]$ 
4  else let  $T[1..n]$  be a new array
5       $q = \lfloor (p + r) / 2 \rfloor$ 
6       $q' = q - p + 1$ 
7      spawn P-MERGE-SORT( $A, p, q, T, 1$ )
8      P-MERGE-SORT( $A, q + 1, r, T, q' + 1$ )
9      sync
10     P-MERGE( $T, 1, q', q' + 1, n, B, s$ )
```

P-MERGE($T, p_1, r_1, p_2, r_2, A, p_3$)

```

1   $n_1 = r_1 - p_1 + 1$ 
2   $n_2 = r_2 - p_2 + 1$ 
3  if  $n_1 < n_2$                                 // ensure that  $n_1 \geq n_2$ 
4      exchange  $p_1$  with  $p_2$ 
5      exchange  $r_1$  with  $r_2$ 
6      exchange  $n_1$  with  $n_2$ 
7  if  $n_1 == 0$                                 // both empty?
8      return
9  else  $q_1 = \lfloor (p_1 + r_1)/2 \rfloor$ 
10      $q_2 = \text{BINARY-SEARCH}(T[q_1], T, p_2, r_2)$ 
11      $q_3 = p_3 + (q_1 - p_1) + (q_2 - p_2)$ 
12      $A[q_3] = T[q_1]$ 
13     spawn P-MERGE( $T, p_1, q_1 - 1, p_2, q_2 - 1, A, p_3$ )
14     P-MERGE( $T, q_1 + 1, r_1, q_2, r_2, A, q_3 + 1$ )
15     sync
```

BINARY-SEARCH(x, T, p, r)

```

1   $low = p$ 
2   $high = \max(p, r + 1)$ 
3  while  $low < high$ 
4       $mid = \lfloor (low + high)/2 \rfloor$ 
5      if  $x \leq T[mid]$ 
6           $high = mid$ 
7      else  $low = mid + 1$ 
8  return  $high$ 
```

2.3 Time Complexity Analysis

Where n is the number of data points in A , analyze the time complexity of the given algorithm with respect to n . Write the result of your analysis in big- O notation, i.e. $O(n^2)$ in the space below.

.....

2.4 Space Complexity Analysis

Where n is the number of data points in A , analyze the space complexity of the given algorithm with respect to n . Write the result of your analysis in big- O notation, i.e. $O(n \cdot \log(n))$ in the space below.

.....

Name, SID, Date

2.5 Optimize the Algorithm for a Purpose

Choose an optimization, either in time or in space, for the given algorithm. Write your intended big- O notation, i.e. $O(1)$, $O(n)$, etc., in the space below, and write N/A in the other space.

- Time Complexity:
- Space Complexity:

What application would benefit from the purpose of the above optimization? Why? Write two sentences to answer these questions in the space below.

.....
.....
.....
.....

2.6 New Algorithm Design and Implementation

In the space below, design an algorithm that achieves the same purpose of the given algorithm, but includes the optimization you have specified above. Use pseudocode written in a style similar to the given algorithm, and implement it in Java. You may use as many additional pages as necessary for this purpose.

3 What to Turn In

Turn in one PDF or Word document on Blackboard, containing the following items.

1. All pages scanned or photographed of the In Class Assignment completed document.
2. Any additional pages you used to complete the assignment.
3. All code created for the assignment, along with test cases.
4. One statement indicating which parts of your implementation(s) are working, and which parts are not.
5. Screenshots demonstrating the code working, if it is working.