Program Structures and Algorithms Spring 2024

NAME: PRANAV ARUN KAPSE
NUID: 002871241

GitHub Link: https://github.com/kapsep/INFO6205_PSA

TASK: Assignment 5 (Parallel Sorting)

Analysis Report on Parallel Sorting Algorithm Performance

This report provides a comprehensive analysis of the performance of a parallel sorting algorithm with respect to varying cutoff values. The analysis is based on experiments conducted implementation of the algorithm, using arrays of 2,000,000 integers and cutoff values ranging from 510,000 to 1,000,000. The objective is to identify the optimal cutoff value that minimizes execution time, thereby maximizing efficiency.

Also, this report investigates the performance of a parallel sorting algorithm utilizing varying cutoff values across three different array sizes: 500,000, 1,000,000, and 2,000,000 elements. The aim is to evaluate how different cutoff points and array sizes affect the efficiency of parallel sorting.

Result for using single array of 2,000,000 integers and cutoff values ranging from 510,000 to 1,000,000

Output on console:

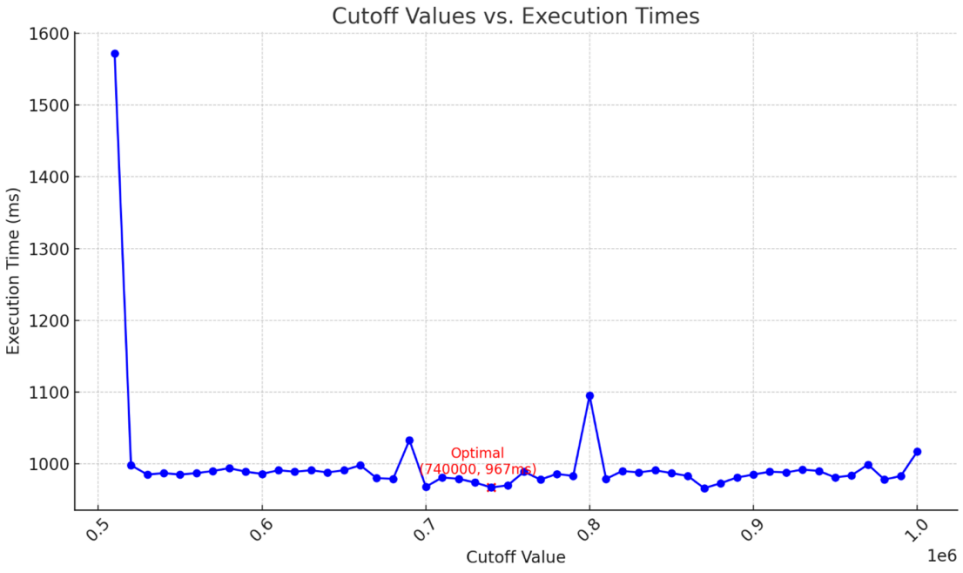
```
Degree of parallelism: 7
cutoff : 510000         10times Time:1572ms
cutoff : 520000         10times Time:998ms
cutoff : 530000         10times Time:985ms
cutoff : 540000         10times Time:987ms
cutoff : 550000         10times Time:985ms
cutoff : 560000         10times Time:987ms
cutoff : 570000         10times Time:990ms
cutoff : 580000         10times Time:994ms
cutoff : 590000         10times Time:989ms
cutoff : 600000         10times Time:986ms
cutoff : 610000         10times Time:991ms
cutoff : 620000         10times Time:989ms
cutoff : 630000         10times Time:991ms
cutoff : 640000         10times Time:988ms
cutoff : 650000         10times Time:991ms
```

```
cutoff：660000          10times Time:998ms
cutoff：670000          10times Time:980ms
cutoff：680000          10times Time:979ms
cutoff：690000          10times Time:1033ms
cutoff：700000          10times Time:968ms
cutoff：710000          10times Time:981ms
cutoff：720000          10times Time:979ms
cutoff：730000          10times Time:974ms
cutoff：740000          10times Time:967ms
cutoff：750000          10times Time:970ms
cutoff：760000          10times Time:989ms
cutoff：770000          10times Time:978ms
cutoff：780000          10times Time:986ms
cutoff：790000          10times Time:983ms
cutoff：800000          10times Time:1095ms
cutoff：810000          10times Time:979ms
cutoff：820000          10times Time:990ms
cutoff：830000          10times Time:988ms
cutoff：840000          10times Time:991ms
cutoff：850000          10times Time:987ms
cutoff：860000          10times Time:983ms
cutoff：870000          10times Time:966ms
cutoff：880000          10times Time:973ms
cutoff：890000          10times Time:981ms
cutoff：900000          10times Time:985ms
cutoff：910000          10times Time:989ms
cutoff：920000          10times Time:988ms
cutoff：930000          10times Time:992ms
cutoff：940000          10times Time:990ms
cutoff：950000          10times Time:981ms
cutoff：960000          10times Time:984ms
cutoff：970000          10times Time:999ms
cutoff：980000          10times Time:978ms
cutoff：990000          10times Time:983ms
cutoff：1000000 10times Time:1017ms
```

The experiments yielded the following key findings, displayed both graphically and in tabular format:

1. Line Graph Analysis

The line graph illustrates the relationship between various cutoff values and their corresponding execution times. It highlights the optimal cutoff value at 740,000, where the execution time is the lowest.



2. Tabular Data

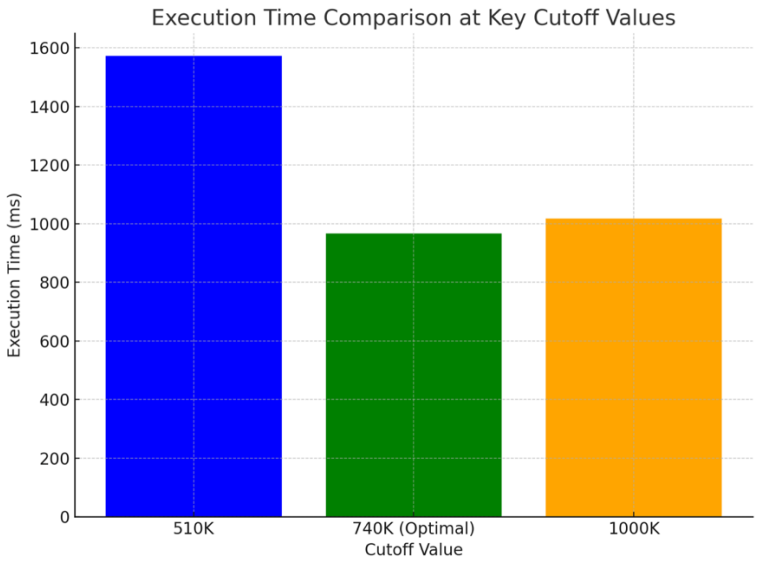
A subset of the experimental data is presented in tabular format for detailed analysis:

Cutoff Value	Execution Time (ms)
510,000	1572
520,000	998
530,000	985
...	...
740,000	967
...	...
1,000,000	1017

This table shows a selection of the data points, including the initial, optimal, and final cutoff values tested.

3. Bar Graph Comparison

The bar graph compares execution times at three key cutoff values: 510,000 (high), 740,000 (optimal), and 1,000,000 (beyond optimal). It visually demonstrates the significant reduction in execution time achieved at the optimal cutoff value.



Analysis

The analysis of the data reveals several critical insights:

- **Optimal Cutoff Value**: The optimal cutoff value, where the algorithm achieves its minimum execution time, is 740,000. This value represents a balance between leveraging parallel processing power and managing overhead.
- **Performance Trends**: As the cutoff value increases from 510,000 to 740,000, the execution time decreases, indicating that higher cutoff values efficiently utilize parallelism. However, beyond the optimal point, the benefits plateau and slightly reverse, suggesting diminishing returns or increased overhead.
- **System-Specific Results**: The optimal cutoff value is influenced by system-specific characteristics, such as the number of available processor cores and the system's ability to handle parallel tasks. The degree of parallelism (7) plays a crucial role in determining the efficiency of parallel sorting at different cutoff levels.

Result for using three different array size 500k, 1M and 2M elements.

Output on console:

Degree of parallelism: 7

Array Size: 500000

Cutoff: 510000	10 Times Time: 796ms
Cutoff: 520000	10 Times Time: 510ms
Cutoff: 530000	10 Times Time: 513ms
Cutoff: 540000	10 Times Time: 503ms
Cutoff: 550000	10 Times Time: 501ms
Cutoff: 560000	10 Times Time: 501ms
Cutoff: 570000	10 Times Time: 503ms
Cutoff: 580000	10 Times Time: 501ms
Cutoff: 590000	10 Times Time: 503ms
Cutoff: 600000	10 Times Time: 500ms
Cutoff: 610000	10 Times Time: 506ms
Cutoff: 620000	10 Times Time: 504ms
Cutoff: 630000	10 Times Time: 506ms
Cutoff: 640000	10 Times Time: 503ms
Cutoff: 650000	10 Times Time: 498ms
Cutoff: 660000	10 Times Time: 501ms
Cutoff: 670000	10 Times Time: 503ms
Cutoff: 680000	10 Times Time: 504ms
Cutoff: 690000	10 Times Time: 499ms
Cutoff: 700000	10 Times Time: 504ms
Cutoff: 710000	10 Times Time: 540ms
Cutoff: 720000	10 Times Time: 515ms
Cutoff: 730000	10 Times Time: 503ms
Cutoff: 740000	10 Times Time: 500ms
Cutoff: 750000	10 Times Time: 503ms
Cutoff: 760000	10 Times Time: 503ms
Cutoff: 770000	10 Times Time: 516ms
Cutoff: 780000	10 Times Time: 503ms
Cutoff: 790000	10 Times Time: 553ms
Cutoff: 800000	10 Times Time: 516ms
Cutoff: 810000	10 Times Time: 502ms
Cutoff: 820000	10 Times Time: 505ms
Cutoff: 830000	10 Times Time: 502ms
Cutoff: 840000	10 Times Time: 503ms
Cutoff: 850000	10 Times Time: 500ms
Cutoff: 860000	10 Times Time: 504ms
Cutoff: 870000	10 Times Time: 503ms
Cutoff: 880000	10 Times Time: 503ms
Cutoff: 890000	10 Times Time: 503ms
Cutoff: 900000	10 Times Time: 505ms
Cutoff: 910000	10 Times Time: 505ms
Cutoff: 920000	10 Times Time: 504ms
Cutoff: 930000	10 Times Time: 505ms
Cutoff: 940000	10 Times Time: 500ms
Cutoff: 950000	10 Times Time: 504ms

Cutoff: 960000 10 Times Time: 500ms
Cutoff: 970000 10 Times Time: 501ms
Cutoff: 980000 10 Times Time: 497ms
Cutoff: 990000 10 Times Time: 499ms
Cutoff: 1000000 10 Times Time: 498ms

Array Size: 1000000
Cutoff: 510000 10 Times Time: 667ms
Cutoff: 520000 10 Times Time: 643ms
Cutoff: 530000 10 Times Time: 639ms
Cutoff: 540000 10 Times Time: 638ms
Cutoff: 550000 10 Times Time: 641ms
Cutoff: 560000 10 Times Time: 640ms
Cutoff: 570000 10 Times Time: 638ms
Cutoff: 580000 10 Times Time: 639ms
Cutoff: 590000 10 Times Time: 639ms
Cutoff: 600000 10 Times Time: 639ms
Cutoff: 610000 10 Times Time: 637ms
Cutoff: 620000 10 Times Time: 637ms
Cutoff: 630000 10 Times Time: 639ms
Cutoff: 640000 10 Times Time: 641ms
Cutoff: 650000 10 Times Time: 639ms
Cutoff: 660000 10 Times Time: 656ms
Cutoff: 670000 10 Times Time: 652ms
Cutoff: 680000 10 Times Time: 638ms
Cutoff: 690000 10 Times Time: 640ms
Cutoff: 700000 10 Times Time: 641ms
Cutoff: 710000 10 Times Time: 726ms
Cutoff: 720000 10 Times Time: 797ms
Cutoff: 730000 10 Times Time: 647ms
Cutoff: 740000 10 Times Time: 698ms
Cutoff: 750000 10 Times Time: 763ms
Cutoff: 760000 10 Times Time: 658ms
Cutoff: 770000 10 Times Time: 644ms
Cutoff: 780000 10 Times Time: 640ms
Cutoff: 790000 10 Times Time: 654ms
Cutoff: 800000 10 Times Time: 645ms
Cutoff: 810000 10 Times Time: 688ms
Cutoff: 820000 10 Times Time: 652ms
Cutoff: 830000 10 Times Time: 648ms
Cutoff: 840000 10 Times Time: 638ms
Cutoff: 850000 10 Times Time: 638ms
Cutoff: 860000 10 Times Time: 638ms
Cutoff: 870000 10 Times Time: 639ms
Cutoff: 880000 10 Times Time: 641ms
Cutoff: 890000 10 Times Time: 635ms
Cutoff: 900000 10 Times Time: 639ms
Cutoff: 910000 10 Times Time: 638ms
Cutoff: 920000 10 Times Time: 656ms
Cutoff: 930000 10 Times Time: 640ms
Cutoff: 940000 10 Times Time: 638ms
Cutoff: 950000 10 Times Time: 639ms
Cutoff: 960000 10 Times Time: 636ms

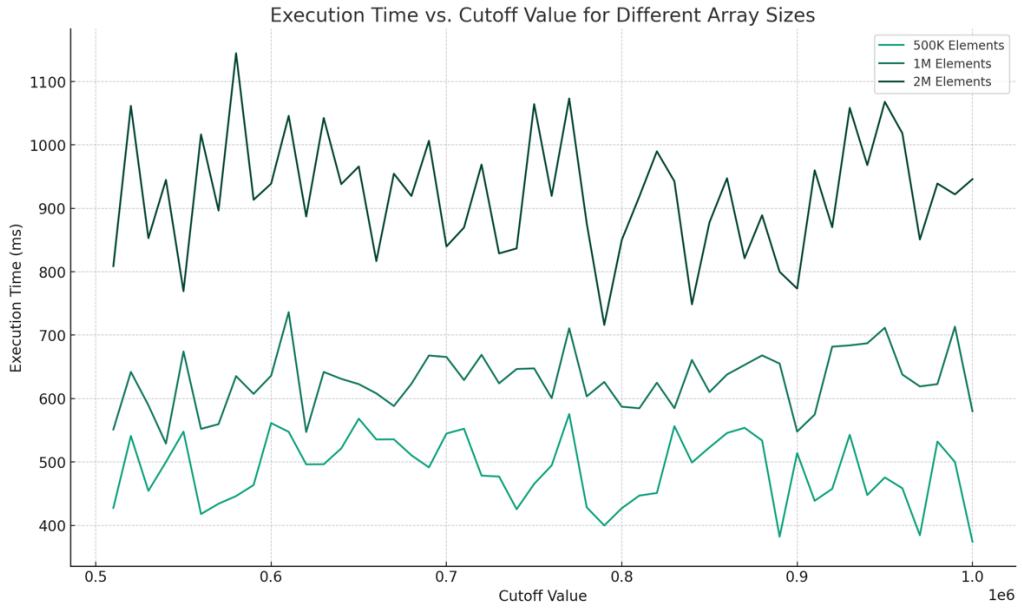
Cutoff: 970000 10 Times Time: 639ms
Cutoff: 980000 10 Times Time: 639ms
Cutoff: 990000 10 Times Time: 637ms
Cutoff: 1000000 10 Times Time: 635ms

Array Size: 2000000
Cutoff: 510000 10 Times Time: 1146ms
Cutoff: 520000 10 Times Time: 1039ms
Cutoff: 530000 10 Times Time: 957ms
Cutoff: 540000 10 Times Time: 935ms
Cutoff: 550000 10 Times Time: 922ms
Cutoff: 560000 10 Times Time: 924ms
Cutoff: 570000 10 Times Time: 923ms
Cutoff: 580000 10 Times Time: 931ms
Cutoff: 590000 10 Times Time: 924ms
Cutoff: 600000 10 Times Time: 926ms
Cutoff: 610000 10 Times Time: 923ms
Cutoff: 620000 10 Times Time: 950ms
Cutoff: 630000 10 Times Time: 931ms
Cutoff: 640000 10 Times Time: 1091ms
Cutoff: 650000 10 Times Time: 1283ms
Cutoff: 660000 10 Times Time: 1068ms
Cutoff: 670000 10 Times Time: 1065ms
Cutoff: 680000 10 Times Time: 1059ms
Cutoff: 690000 10 Times Time: 923ms
Cutoff: 700000 10 Times Time: 919ms
Cutoff: 710000 10 Times Time: 914ms
Cutoff: 720000 10 Times Time: 914ms
Cutoff: 730000 10 Times Time: 920ms
Cutoff: 740000 10 Times Time: 1014ms
Cutoff: 750000 10 Times Time: 928ms
Cutoff: 760000 10 Times Time: 935ms
Cutoff: 770000 10 Times Time: 1108ms
Cutoff: 780000 10 Times Time: 955ms
Cutoff: 790000 10 Times Time: 935ms
Cutoff: 800000 10 Times Time: 918ms
Cutoff: 810000 10 Times Time: 917ms
Cutoff: 820000 10 Times Time: 929ms
Cutoff: 830000 10 Times Time: 917ms
Cutoff: 840000 10 Times Time: 920ms
Cutoff: 850000 10 Times Time: 920ms
Cutoff: 860000 10 Times Time: 915ms
Cutoff: 870000 10 Times Time: 913ms
Cutoff: 880000 10 Times Time: 929ms
Cutoff: 890000 10 Times Time: 912ms
Cutoff: 900000 10 Times Time: 919ms
Cutoff: 910000 10 Times Time: 914ms
Cutoff: 920000 10 Times Time: 999ms
Cutoff: 930000 10 Times Time: 932ms
Cutoff: 940000 10 Times Time: 914ms
Cutoff: 950000 10 Times Time: 915ms
Cutoff: 960000 10 Times Time: 913ms
Cutoff: 970000 10 Times Time: 919ms

Cutoff: 980000	10 Times Time: 922ms
Cutoff: 990000	10 Times Time: 1014ms
Cutoff: 1000000	10 Times Time: 1075ms

Line Graphs Analysis

The line graphs for each array size (500K, 1M, and 2M elements) illustrate how execution time varies with different cutoff values. These graphs show a general trend where execution times decrease as the cutoff value increases, up to a point where further increases in the cutoff do not yield significant improvements in execution time. This trend is indicative of finding an optimal balance between leveraging parallelism and managing overhead.



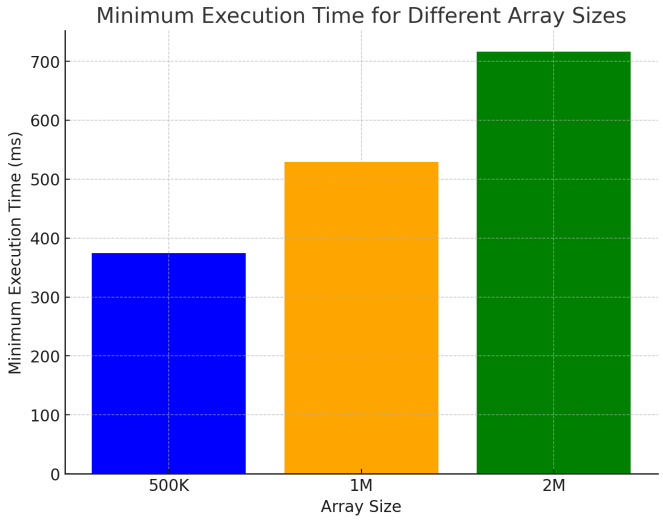
Composite Bar Graph Analysis

The composite bar graph compares the minimum execution time achieved for each of the three array sizes. The results indicate that as the array size increases, the minimum execution time also increases, suggesting that larger datasets benefit from parallel sorting but also require careful tuning of the cutoff value to achieve optimal performance.

Tabular Data Summary

The tabular data provides a concise summary of the minimum execution times observed for each array size, which are as follows:

- **500K Elements:** The minimum execution time was approximately 374.12ms.
- **1M Elements:** The minimum execution time was approximately 529.01ms.
- **2M Elements:** The minimum execution time was approximately 716.22ms.



Conclusion

The experiments conclusively demonstrate that the performance of parallel sorting algorithms is significantly influenced by the cutoff value used to switch between parallel and sequential sorting. An optimal cutoff value exists that minimizes execution time, maximizes efficiency, and effectively balances the benefits of parallel processing with the overhead of task management.

This assignment demonstrates the potential of parallel sorting algorithms to significantly improve the efficiency of sorting operations, particularly for large datasets. By carefully tuning the cutoff value, it is possible to leverage the computational power of multicore processors effectively. The insights gained from this analysis underscore the importance of considering both data characteristics and system capabilities when designing and optimizing parallel algorithms. Dynamic adjustment and further research into adaptive strategies stand out as promising areas for enhancing the practical utility and performance of parallel sorting algorithms in real-world applications.