Lớp: KHTN24

**BÁO CÁO KẾT QUẢ THỬ NGHIỆM**

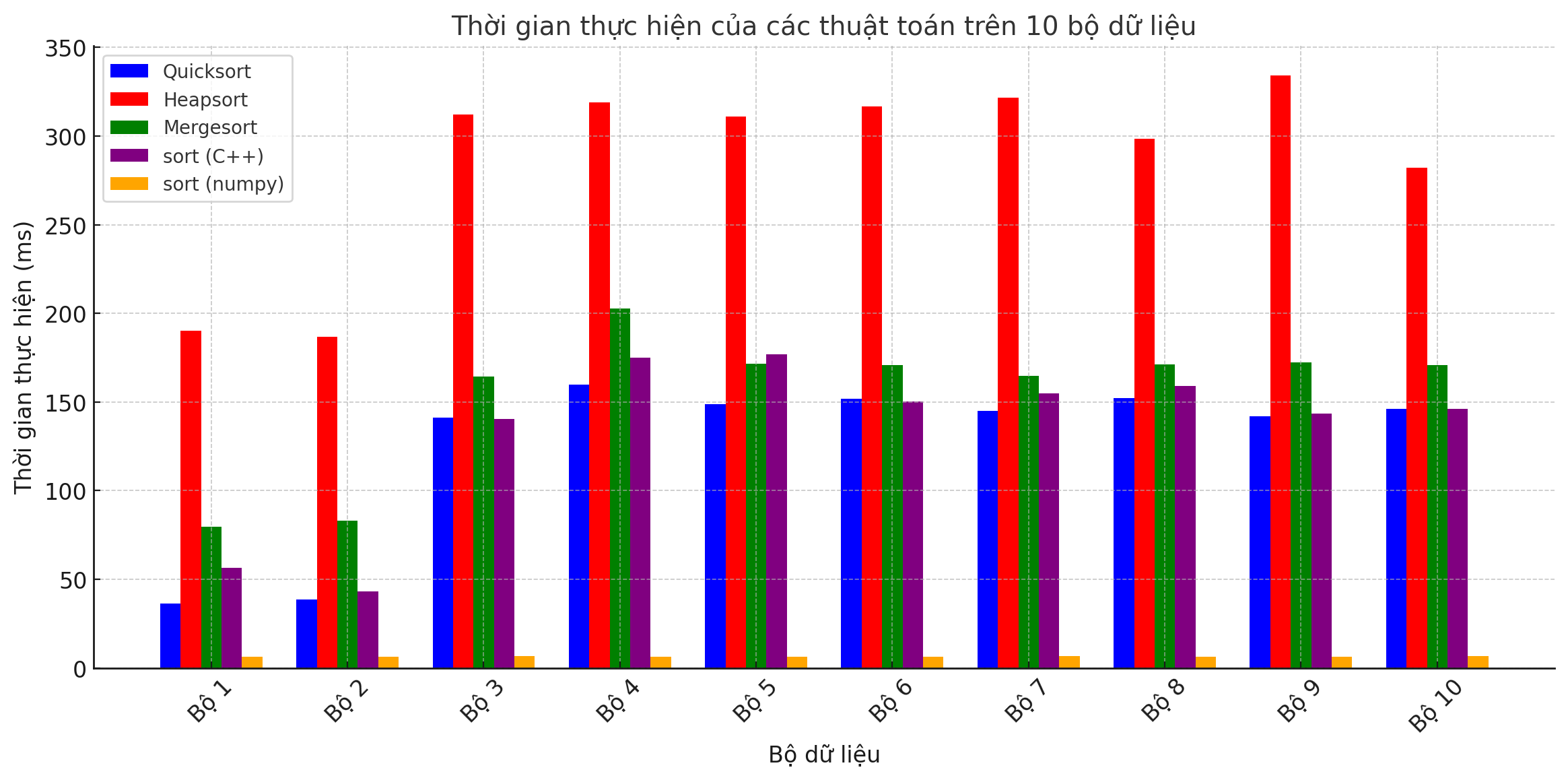
**Sinh viên thực hiện: Lê Thiên Quân**

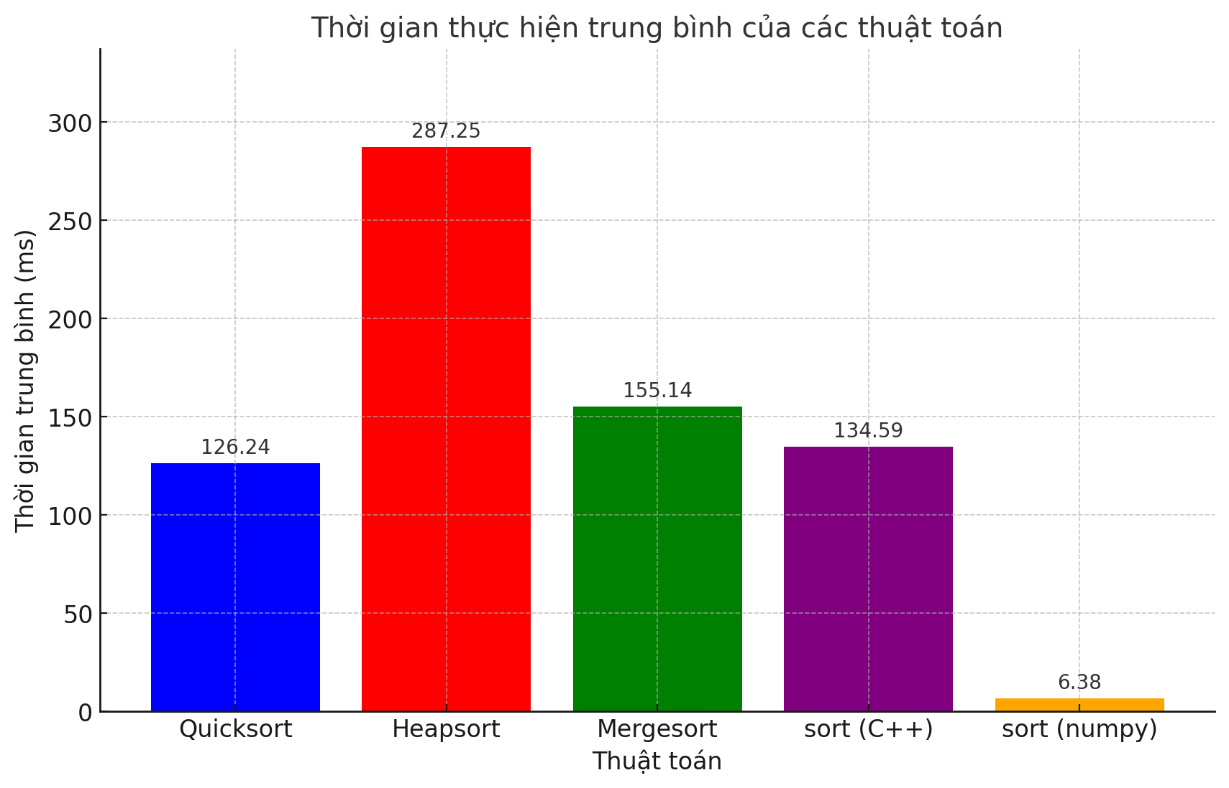
**Nội dung báo cáo: Báo cáo về thời gian thực hiện của các thuật toán sắp xếp dựa trên bộ dữ liệu tự tạo.**

1. ***Kết quả thử nghiệm***
   1. ***Bảng thời gian thực hiện***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dữ liệu** | **Thời gian thực hiện (ms)** | | | | |
| **Quicksort** | **Heapsort** | **Mergesort** | **sort (C++)** | **sort (numpy)** |
| 1 | 36.4204 | 190.285 | 79.7168 | 56.3538 | 6.385326 |
| 2 | 38.7175 | 186.946 | 82.9820 | 43.3276 | 6.185532 |
| 3 | 141.052 | 312.033 | 164.307 | 140.349 | 6.603003 |
| 4 | 159.825 | 318.983 | 202.779 | 175.167 | 6.378174 |
| 5 | 148.915 | 311.077 | 171.588 | 176.827 | 6.312847 |
| 6 | 152.005 | 316.662 | 170.659 | 150.190 | 6.253004 |
| 7 | 145.099 | 321.673 | 166.394 | 154.914 | 6.582499 |
| 8 | 152.240 | 298.384 | 171.288 | 158.958 | 6.357431 |
| 9 | 141.954 | 334.272 | 172.477 | 143.484 | 6.201029 |
| 10 | 146.175 | 282.228 | 170.862 | 146.281 | 6.571770 |
| Trung bình | 126.240 | 287.254 | 155.305 | 134.585 | 6.383061 |

* 1. ***Biểu đồ (cột) thời gian thực hiện***

******

******

1. ***Kết luận:***

**- sort (numpy):** Nhanh nhất trong tất cả các thuật toán, với thời gian thực thi cực kỳ nhỏ (~6.38ms). Điều này là do numpy.sort() sử dụng thuật toán Timsort (kết hợp Merge Sort và Insertion Sort) và được tối ưu hóa bằng cách chạy trên C-level, tận dụng hiệu suất của NumPy.

**-** **Quicksort:** Có thời gian thực thi khá tốt (~126.24ms), ổn định và hiệu quả cho hầu hết các bộ dữ liệu.

**-** **Mergesort**: Mặc dù có độ phức tạp O(n log n) giống Quicksort, nhưng thường chậm hơn (~155.14ms), do tốn bộ nhớ cho quá trình trộn và ít tối ưu hơn trên bộ dữ liệu nhỏ.

- **sort (C++):** Có hiệu suất gần giống Quicksort (~134.59ms), cho thấy C++ STL std::sort() có sự tối ưu hóa mạnh mẽ.

- **Heapsort:** Chậm nhất (~287.25ms), do nhiều phép so sánh và hoán đổi, dù có độ phức tạp O(n log n). Heapsort ít được sử dụng trong thực tế vì không tối ưu bằng QuickSort hoặc MergeSort.