Lớp: IT003.P21.CTTN

**BÁO CÁO KẾT QUẢ THỰC NGHIỆM CÁC GIẢI THUẬT SẮP XẾP NỘI**

Thời gian thực hiện: 011/03 – 16/03/2022

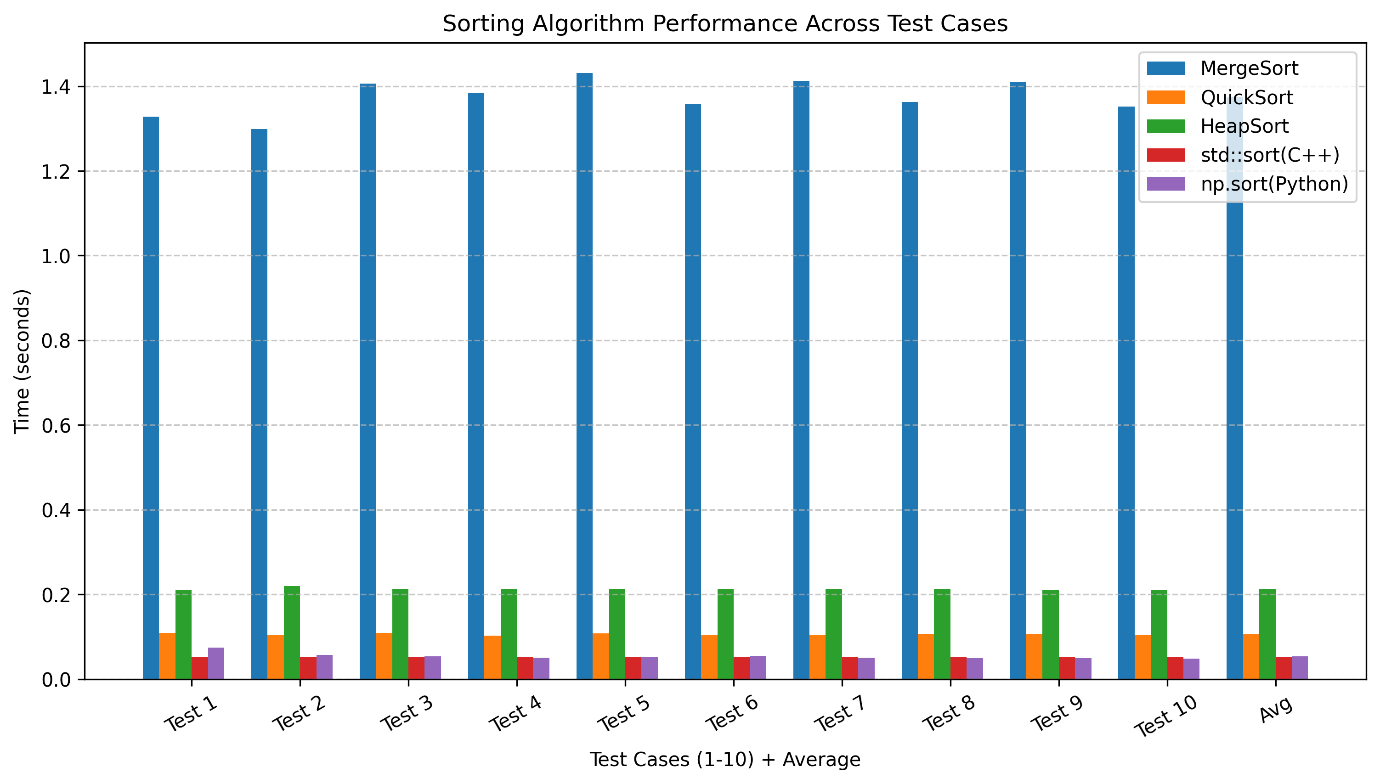
**Sinh viên thực hiện: Đoàn Quốc Kiên. MSSV: 24520879**

**Nội dung báo cáo:**

1. ***Kết quả thử nghiệm***
   1. ***Bảng thời gian thực hiện[[1]](#footnote-1)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dữ liệu** | **Thời gian thực hiện (ms)** | | | | |
| **MergeSort** | **QuickSort** | **HeapSort** | **sort (C++)** | **sort (numpy)** |
| 1 | 1327.48 | 108.759 | 210.891 | 51.772 | 74.954 |
| 2 | 1298.4 | 104.109 | 219.548 | 51.5381 | 57.509 |
| 3 | 1406.33 | 109.205 | 211.689 | 51.6174 | 54.006 |
| 4 | 1383.99 | 103.159 | 211.711 | 52.3944 | 50.615 |
| 5 | 1431.28 | 108.334 | 211.991 | 52.302 | 51.654 |
| 6 | 1358.09 | 104.47 | 211.753 | 51.9111 | 54.41 |
| 7 | 1412.18 | 103.65 | 211.959 | 51.832 | 50.735 |
| 8 | 1362.89 | 106.464 | 211.847 | 51.6908 | 49.511 |
| 9 | 1408.73 | 105.89 | 211.2 | 51.6505 | 49 |
| 10 | 1351.76 | 105.062 | 210.4 | 51.6836 | 48.513 |
| Trung bình | 1374.11 | 105.91 | 212.299 | 51.8392 | 54.091 |

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



1. ***Kết luận:***

Từ kết quả thử nghiệm trên tập dữ liệu lớn, ta có thể đưa ra một số nhận xét quan trọng về hiệu suất của các thuật toán sắp xếp:

- Hiệu suất tổng thể:

+ sort() của C++ là thuật toán nhanh nhất, với thời gian trung bình 51.8392 ms. Điều này là do C++ sử dụng thuật toán Introsort, kết hợp giữa QuickSort, HeapSort và Insertion Sort để tối ưu tốc độ.

+ sort() của NumPy xếp thứ hai (54.091 ms), cho thấy hiệu suất rất cao do sử dụng thuật toán Timsort, một phiên bản cải tiến của MergeSort.

- So sánh các thuật toán truyền thống:

+ **QuickSort** có hiệu suất tốt thứ ba (**105.91 ms**), nhanh hơn đáng kể so với HeapSort (**212.299 ms**) nhưng vẫn chậm hơn sort() của C++ và NumPy. Điều này cho thấy rằng QuickSort hoạt động tốt nhưng chưa tối ưu bằng các thuật toán sắp xếp lai (hybrid).

+ **MergeSort** là thuật toán chậm nhất (**1374.11 ms**), do độ phức tạp cao và lượng bộ nhớ bổ sung cần thiết cho việc chia nhỏ mảng.

- Đánh giá tổng quan:

+ Các thuật toán có sẵn trong thư viện chuẩn (C++ STL, NumPy) đều vượt trội so với các thuật toán thủ công.

+ HeapSort là lựa chọn tốt hơn so với MergeSort khi cần đảm bảo hiệu suất ổn định.

+ QuickSort có thể cải thiện hiệu suất nếu áp dụng chiến lược chọn pivot tối ưu hơn.

- Kết luận cuối cùng:

Việc sử dụng các thuật toán sắp xếp **có sẵn trong thư viện chuẩn** như std::sort() của C++ hoặc numpy.sort() trong Python là lựa chọn tối ưu cho các ứng dụng thực tế. Nếu cần một thuật toán có độ ổn định cao, **QuickSort là lựa chọn tốt hơn MergeSort**. Trong khi đó, HeapSort có thể được cân nhắc khi cần đảm bảo thời gian thực thi ổn định mà không phụ thuộc nhiều vào dữ liệu đầu vào.

1. ***Thông tin chi tiết –*** [***link github***](https://github.com/DoanQuocKien/SortingReport)

1. Số liệu chỉ mang tính minh họa [↑](#footnote-ref-1)