Lớp: KHTN 2023

**BÁO CÁO KẾT QUẢ THỬ NGHIỆM**

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

**Sinh viên thực hiện: Nguyễn Thiên Bảo ( MSSV: 23520127 )**

**Nội dung báo cáo:**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dữ liệu** | **Thời gian thực hiện (s)** | | | | |
| **Quicksort** | **Heapsort** | **Mergesort** | **sort (C++)** | **sort (numpy)** |
| 1 | 1.7971 | 4.1078 | 2.2576 | 0.1135 | 0.0108 |
| 2 | 1.7504 | 4.1253 | 2.1412 | 0.0875 | 0.0195 |
| 3 | 2.2286 | 8.3425 | 2.9681 | 0.2531 | 0.0687 |
| 4 | 2.2262 | 8.1732 | 2.9634 | 0.2125 | 0.0703 |
| 5 | 2.1998 | 9.3715 | 2.9139 | 0.2492 | 0.0753 |
| 6 | 2.3564 | 9.3563 | 2.9932 | 0.2243 | 0.0882 |
| 7 | 2.2408 | 8.4537 | 2.9324 | 0.2094 | 0.0657 |
| 8 | 2.2676 | 8.3939 | 2.9195 | 0.2145 | 0.0656 |
| 9 | 2.3124 | 8.593 | 2.9195 | 0.2193 | 0.0656 |
| 10 | 2.4043 | 7.0559 | 2.9664 | 0.2097 | 0.0629 |
| Trung bình | 2.1784 | 7.5973 | 2.7975 | 0.1993 | 0.0593 |

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

1. ***Kết luận:***

- Cả Quick sort, Heap sort và Merge sort đều có độ phức tạp là O(nlogn) . Nhưng giữa chúng đều có hiệu suất khác biệt theo bộ dữ liệu 10 file:

+Quick sort < Merge sort < Heap sort đối với thời gian chạy.

+Quick sort, Heap sort < Merge sort đối với bộ nhớ.

+ Quicksort:

* + Trong trường hợp trung bình, Quick sort thường có thời gian thực hiện nhanh nhất giữa ba thuật toán với bộ dữ liệu lớn. Tuy nhiên, trong trường hợp xấu nhất, khi pivot được chọn không phù hợp, quick sort có thể có hiệu suất rất tồi tệ ( độ phức tạp O(n^2)). Nhưng có lẽ vì ở 10 file có bộ dữ liệu khá lớn nên khả năng pivot random chọn vị trí để chạy khá tốt và khá ổn định ( nhưng chưa ổn định bằng Merge sort).
  + Quick sort cần ít bộ nhớ hơn Merge sort do không cần lưu trữ các phần tử trong một mảng phụ để hợp nhất hoặc trong một heap.

+ Merge sort:

* + Merge sort có hiệu suất ổn định (O(nlogn)) trong mọi trường hợp và không bị ảnh hưởng bởi việc pivot được chọn như Quick sort. Có thể coi là ổn định nhất trong ba thuật toán.
  + Tuy nhiên, merge sort cần một mảng phụ bổ sung bằng kích thước bằng với mảng ban đầu để thực hiện quá trình hợp nhất. Điều này tăng bộ nhớ sử dụng của nó so với Quick sort và Heap sort.

+ Heap sort:

* + Heap sort trên lý thuyết là có độ phức tạp O(nlogn) nhưng dường như thời gian chạy của Heap sort rất chậm (dường như gấp đôi, gấp ba Merge sort và Quick sort với bộ dữ liệu 10 file ) vì cần duy trì cấu trúc Heap trong quá trình sắp xếp. Heap sort còn khá không ổn định (theo biểu đồ), vì với mỗi bước, nó luôn phải thay đổi thứ tự tương đối của các phần tử để giữ vững cấu trúc Heap.
  + Heap sort sử dụng một heap như một cấu trúc dữ liệu phụ, do đó không cần một mảng bổ sung như Merge sort. Tuy nhiên, cấu trúc Heap này cũng tốn bộ nhớ.

-Sort (C++) và Sort (numpy) đều được thiết kế với những thuật toán phức tạp, ổn định và hiệu quả ( ví dụ như Sort (C++) dùng Introsort , Python dùng Tim sort ).

+Với bộ dữ liệu 10 file , dường như Sort (numpy) nhanh hơn Sort (C++). Lý do là trong một số trường hợp, numpy.sort có thể nhanh hơn do việc tối ưu hóa đặc biệt cho các hoạt động trên mảng, với các mảng lớn chứa các kiểu dữ liệu số học, np.sort của NumPy có thể nhanh hơn nhờ sự tối ưu hóa cao cho các phép tính số và tính song song trên mảng. NumPy cũng cung cấp nhiều hàm xử lý mảng khác rất hiệu quả. Còn Sort(C++) thường hoạt động “in-place”-không yêu cầu không gian bổ sung, điều này có thể làm tăng hiệu suất khi làm việc với các mảng lớn với dữ liệu phức tạp.

+Đối với dữ liệu nguyên thủy (int, float,...) và mảng nhỏ thì Sort(C++) “có thể” nhanh hơn Sort(Numpy), còn với bộ dữ liệu lớn với dữ liệu phức tạp thì Sort(numpy) có lợi thế hơn theo một số nguồn em tham khảo :

[Why is Python sort faster than that of C++ - Stack Overflow](https://stackoverflow.com/questions/29324660/why-is-python-sort-faster-than-that-of-c)

**=>** Sort (numpy) < Sort (C++) < Quick sort < Merge sort < Heap sort về thời gian chạy đối với bộ dữ liệu 10 file em đã tạo. Tuy nhiên đó chỉ là thống kê của riêng em (Vì những thuật toán em tự viết và dữ liệu tự tạo nên đôi khi có những chênh lệnh ), hiệu quả thực tế có thể thay đổi tùy thuộc vào nhiều yếu tố khác nhau.

1. ***Thông tin chi tiết – link github***

Báo cáo, mã nguồn, chạy thử và bộ dữ liệu thử nghiệm đều được em upload tại link github sau:

https://github.com/Limdim1604/IT003/tree/main/SORT-RUNNING-TIME