Lớp: KHTN 2023

**BÁO CÁO KẾT QUẢ THỬ NGHIỆM**

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

**Sinh viên thực hiện: Nguyễn Đình Thiên Quang - 23521285**

**Nội dung báo cáo: So sánh tốc độ của các thuật toán sắp xếp**

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 | 13.3279 | 39.668 | 40.2582 | 45.911 | 9.0983 |
| 2 | 13.3405 | 49.6342 | 41.6861 | 9.9983 | 8.0083 |
| 3 | 32.5444 | 57.3567 | 64.7123 | 31.6627 | 15.00806 |
| 4 | 30.3233 | 80.6763 | 53.334 | 31.6654 | 15.55669 |
| 5 | 32.9662 | 133.011 | 53.3321 | 31.6717 | 15.0044 |
| 6 | 31.0122 | 63.994 | 53.2822 | 31.6706 | 16.6855 |
| 7 | 29.2044 | 64.6814 | 54.9881 | 30.0975 | 15.1872 |
| 8 | 30.0083 | 57.9123 | 53.3102 | 31.4149 | 15.6509 |
| 9 | 33.4969 | 66.5439 | 50.6629 | 31.658 | 16.6489 |
| 10 | 28.3349 | 56.1346 | 52.4054 | 33.2201 | 16.6758 |
| Trung bình | 27.4559 | 66.96124 | 51.79715 | 30.89702 | 14.3524 |

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

1. ***Kết luận:***
   1. ***Về mặt trung bình: sort của numpy cho thời gian chạy ngắn nhất và gần như tối ưu, sau đó là quick sort và c++ sort, còn heap sort và merge sort có tốc độ thực thi khá chậm (chậm hơn từ 30 -> 50% so với quick sort hay c++ sort, chậm hơn gấp 2~2.5 lần so với numpy sort)***
   2. ***Giải thích lí do:***
      * *Bộ dữ liệu được sinh gồm 10 test 1 triệu số, tất cả các số đều là số thực và sinh ngẫu nhiên trong khoảng từ 0 tới 1 với độ chính xác 6 chữ số sau dấu phẩy*
      * *Trên lí thuyết, tốc độ của quick sort, merge sort, heap sort phải cách nhau không nhiều (vì cùng là O(nlogn)), nhưng bộ dữ liệu được sinh đã bị bias (thiên vị), cụ thể:*

***+ Pivot được lấy random trong quicksort gần như không bao giờ lấy trúng trường hợp tệ nhất (worst-case), theo nghĩa là pivot không bao giờ là phần tử lớn nhất hoặc nhỏ nhất => Tốc độ nhanh và ổn định => Do bộ dữ liệu được sinh theo phương pháp sinh ngẫu nhiên phân bố đều, do đó nếu chọn pivot ở giữa (theo như code) thì xác suất trúng trường hợp tệ nhất là cực kỳ thấp.***

***+ Mặt khác, cấu trúc được sử dụng trong merge sort và heap sort có liên quan tới đệ quy, cũng như sử dụng nhiều cấu trúc con trỏ, đồng thời, thao tác cập nhập (được cài đặt theo Floyd trong trường hợp của Heap sort, hay như với merge sort là thao tác Merge ) có tốc độ xử lý chậm (tệ nhất là O(n)), do đó ảnh hưởng khá lớn tốc độ thuật toán.***

***+ Với c++ sort,thư viện STL thực chất sử dụng thuật toán Intro Sort, là một thuật toán lai (hybrid) giữa quick sort,heap sort, và insertion sort , bằng việc phân tích rất cụ thể dữ liệu thì stl sẽ quyết định nên dùng thuật toán nào để sort, nhưng bộ dữ liệu được sinh ra khá bias, vì thao tác so sánh giữa các số tốn nhiều thời gian, cũng như có nhiều đoạn dữ liệu gây nhiễu (chẳng hạn như nhiều số giống nhau liên tiếp hoặc là sai số rất nhỏ để so sánh).***

***+ Với numpy sort, thực chất là sủ dụng thuật toán lai (hybrid) giữa insertion sort và quick sort, được gọi là tim sort, thuật toán này được thiết kế với độ ổn định cao, cộng thêm các thao tác đọc, so sánh dữ liệu của numpy được thiết kế nhanh hơn phần nào so với c++ nên kết quả cho ra là tốt nhất***

1. ***Thông tin chi tiết – link github, trong repo gibub cần có***
   1. Báo cáo
   2. Mã nguồn
   3. Dữ liệu thử nghiệm