**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**16CNTN**

BÁO CÁO

SO SÁNH CÁC THUẬT TOÁN SẮP XẾP

* Môn : Thực hành cấu trúc dữ liệu và giải thuật
* MSSV : 1612052
* Họ tên : Phạm Minh Chiến
* Lớp : 16CNTN
* Chủ đề : ***So sánh các thuật toán sắp xếp***

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Khoa Công nghệ Thông tin

Đại học Khoa học Tự nhiên TP HCM

Tháng 10/2017

**BÁO CÁO**

**SO SÁNH CÁC THUẬT TOÁN SẮP XẾP**

1. **Các thuật toán cần so sánh, và phương thức so sánh**

* So sánh các thuật toán:
  + Interchange Sort
  + Insertion Sort
  + Quick Sort
  + Heap Sort
  + Radix Sort
  + Flash Sort
* Phương thức so sánh: đo thời gian chạy các thuật với số lượng mảng phát sinh ngẫu nhiên tăng dần
* Các bộ dữ liệu đầu vào là mảng số nguyên N phần tử cấp phát động được phát sinh ngẫu nhiên với số lượng phần tử lần: N = 5000\*k với k =1, 2, 3, ... ,400. Tức 400 bộ dữ liệu đầu vào với bộ lớn nhất là 2000000 số nguyên 4 byte. Tất cả được lưu vào file .txt để đảm bảo tính công bằng khi so sánh các thuật toán. Vì dung lượng của 400 bộ dữ liệu quá lớn (~2GB) nên em không nộp kèm theo Source được, thầy có thể xem ở link này: [www.fshare.vn/folder/J9DTPTPRHTHM](https://www.fshare.vn/folder/J9DTPTPRHTHM)
* Đánh giá ban đầu:

+ Như được biết 6 thuật toán nêu trên có thể chia thành 3 nhóm tương đương với độ phức tạp của nó lần lượt là Nhóm 1 O(n2) gồm có Interchange sort và Insertion sort; Nhóm thứ 2 độ phức tập O(nlogn) có thuật toán Quick Sort, Heap Sort và nhóm 3 là O(n) với 2 thuật toán Radix sort và Flashsort.

+ Với nhóm thuật toán 1 (O(n2)) thì dễ dàng đoán được sự chậm chạp của thuật toán với các bộ dữ liệu lớn.

+ Còn 2 nhóm 2 và 3 thì còn nhiều vần đề nan giải, thoạt nhìn thì có thể thấy nhóm 3 thể hiện tốc độ vượt trội (O(n)) so với O(nlogn). Tuy nhiên nhóm thuật toán 3 cơ bản khác với các thuật toán còn lại của nhóm 1 và 2 ở cách thức sắp xếp, không dựa vào giá trị của 2 giá trị khi so sánh và xác định vị trí của các phần tử. Cả Radix sort và Flash sort đều sử dụng cách tiếp cận là “phân lô” để sắp xếp dữ liệu. Tuy nhiên hiệu quả của nó như thế nào, có tốt so với các thuật toán kia và trong trường hợp nào thì hãy xem kết quả do thời gian chạy các thuật toán trên cùng 1 bộ dữ liệu giống nhau, trên cùng 1 máy tính.

1. **Kết quả thực nghiệm**

* Bảng thời thống kê thời gian chạy các thuật toán:

Bảng thống kê thời gian chạy cho từng thuật toán nằm trong file Excel tên “***1612052 – Thong ke thoi gian chay.xlsx***” trong cùng thư mục.

* *Biểu đồ so sánh các thuật toán:*

Từ biểu đồ trên ta thấy rằng: nết xét các bộ test có số lượng phần tử nhỏ thì thuật toán nhón O(n2) không có mấy khac biệt so với nhóm còn lại. Sự khác biệt rõ ràng hơn khi các bộ dữ liệu có kích thước tăng dần, thoi gian chạy của Interchange sort và Insertion sort tăng lên rất nhanh từ vài giây hàng trăm giây trong khi các thuật toán cao cấp vẫn chỉ khoảng 1, 2 giây. Từ đố ta thấy được chênh lệch đến kinh ngạc của nhóm thuật toán O(n2) so với nhóm thuật toán có độ phức tạp O(nlogn) và O(n). Thời gian chạy của nhóm này tăng chóng mặt, độ dốc của 2 đường biểu diễn đủ cho thấy tính hiệu quả của thuật toán Interchange Sort và Insertion Sort là như thế nào trong thực tế (lượng sắp xếp lớn). Hơn nữa, thuật toán chỉ chấp nhận “chờ được” ở các mảng 5000 đến 500000 phần tử, còn các bộ có N lớn hơn thì khó mà chờ nó chạy xong được. Trong khi các thuật toán nhóm 2, 3 chạy nhanh quá nên thời gian chạy không thấy vẽ trên biểu đồ luôn.

* *So sánh các thuật toán có độ phức tạp O(nlogn) và O(n)*

Đồ thì trên biểu diễn thời gian chạy của các thuật toán QuickSort, HeapSort, RadixSort, FlashSort. Nhìn chung thì chúng chạy rất nhanh. Đặc biệt ấn tượng với QuickSort và RadixSort, nhìn 2 đường biểu diễn gần như là dính vào nhau cho thấy tốc độ QuickSort cũng ngang bằng với RadixSort trong 400 bộ dữ liệu ngẫu nhiên này. Còn phần thuật toán HeapSort thì cũng không thể nói là thua kém QuickSort và RadixSort được, bởi vì cũng chạy thành công trên 400 bộ dữ liệu, thời gian chạy cũng nhanh, kém QuickSort khoảng chừng 0,2s trong bộ cuối cùng – 2000000 phần tử. Cuối cùng là FlashSort, khá bất ngờ khi cái tên FlashSort lại tỏ ra không vượt trội như đúng cái tên “Flash”, thuật toán này chỉ hoàn thành được khoảng 2/3 số bộ test (vấn đề là việc lặp lại đệ qui nhiều lần cộng với việc dò tìm phần tử lớn nhất, nhỏ nhất trong mỗi lần đệ qui gây tràn bộ nhớ stack). Hơn nữa, thời gian chạy trong các trường hợp chạy được lại khong nhanh bằng 3 thuật toán phân tích trên.

1. **Nhận xét, đánh giá**

* Qua các phân tích thời gian chạy của các thuật toán trên bản thân em có nhận xét như sau:
  + Các thuật toác sắp xếp cấp thấp chỉ có thể chấp nhận được trong các trường hợp mảng cần sắp có kích thước nhỏ. Nếu phân tích kĩ thì ta có thể thấy trong các trường hợp sắp ít phần tử thì Interchange sort và Insertion sort còn nhanh hơn cả các thuật toán sắp xếp cấp thấp còn lại. Điều đó giải thích vì sao tác giả thuật toán FlashSort đã chọn 1 thuật toán sắp xếp cấp thấp trong giai đoạn sắp các phần tử từng phân vùng (ít phần tử).
  + Việc đo thời gian chạy như trong bài thí nghiệm này tuy không chính xác tuyệt đối nhưng phản ánh phần nào bản chất của độ phức tạp của thuật toán chỉ mang tính chất tương đối. Có rất nhiều yếu tố ảnh hưởng đến thời gian chạy của một thuật toán, đặc biệt là phân bố dữ liệu. Như đã phân tích bên trên, mặc dù FlashSort được chứng minh là có độ phức tạp O(n) (chính xác là O(m.n)) tuy nhiên vẫn chạy chậm hơn QuickSort và HeapSort, trong khi RadixSort lại tương đương QuickSort. Tóm lại tất cả còn tùy vào ngữ cảnh áp dụng.
  + Nói về tính ổn định thì có thể bình chọn cho 2 thuật toán là HeapSort và QuickSort, ổn về thời gian chạy lẫn khả năng xử lí. Cá nhân em không bình chọn cho RadixSort bởi duy nhất một lí do là nó không xử lí tốt trên dữ liệu số thực. Nhưng nếu dữ liệu sắp xếp chỉ là số nguyên thì chắc chắn RadixSort là một lựa chọn tốt.