BÁO CÁO ĐỒ ÁN CÁC THUẬT TOÁN SẮP XẾP

1. Các thuật toán đã cài đặt

a) Selection Sort

- Ý tưởng: tìm phần tử nhỏ nhất trong mảng, đổi chỗ với phần tử ở vị trí đầu tiên. Xét tiếp mảng bỏ qua phần tử đầu tiên đến khi mảng chỉ còn một phần tử.
- Thuật toán:

Bước 1: i=0;
Bước 2: tìm phần tử a[min] nhỏ nhất trong mảng;
Bước 3: hoán vị a[min[và a[i]
Bước 4: nếu i<n thì i=i+1 và lặp lại bước 2. Ngược lại thì dừng thuật toán.

- Đánh giá: ở lượt thứ i có (n-i) phép so sánh, số phép so sánh không phụ thuộc tình trạng ban đầu của dãy. Số phép so sánh n(n-1)/2. Số phép gán: trường hợp tốt nhất là 0, xấu nhất là 3n(n-1)/2. Độ phức tạp của thuật toán: O(n^2).

b) Interchange Sort

- Ý tưởng: tìm các cặp nghịch thế trong dãy và đổi chỗ hai phần tử trong cùng cặp nghịch thế. Lặp lại với các cặp phần tử tiếp theo trong dãy. Cặp phần tử được gọi là nghịch thế khi nó không thỏa thứ tự sắp xếp.
- Thuật toán:

Bước 1: i=0;
Bước 2: j=i+1;
Bước 3: khi j<n, nếu a[j]<a[i] ta đổi chỗ a[i] và a[j], j=j+1;
Bước 4: i=i+1, nếu i<n-1 thì quay lại bước 2, ngược lại thì dừng.

- Đánh giá: số lượng phép so sánh và phép gán không phụ thuộc tình trạng dãy số ban đầu, số phép hoán vị phụ thuộc vào kết quả so sánh. Số phép so sánh: n(n-1)/2. Số phép gán: trường hợp tốt nhất là 0, xấu nhất là n(n-1)/2. Độ phức tạp của thuật toán là O(n^2).

c) Insertion Sort

- Ý tưởng: giả sử có một dãy n phần tử trong đó i phần tử đầu đã được sắp xếp theo thứ tự. Bằng cách chèn phần tử a[i] vào vị trí thích hợp của dãy đã được sắp xếp thỏa a[k-1] < a[i] < a[k] với 1 <= k <i
- Thuật toán:

Bước 1: i=1;
Bước 2: x=a[i] tìm vị trí pos thích hợp trong đoạn a[0] đến a[i-1] để chèn a[i] vào
Bước 3: dời vị trí các phần tử từ a[pos] đến a[i-1] sang phải 1 vị trí để chèn a[i] vào
Bước 4: a[pos]=x
Bước 5: i=i+1, nếu i<n thì lặp lại bước 2, ngược lại thì dừng.

- Đánh giá: giải thuật thực hiện tất cả n-1 vòng lặp, số lượng phép so sánh và phép gán phụ thuộc vào tình trạng dãy ban đầu. Số phép so sánh: trường hợp tốt nhất là (n-1), xấu nhất là n(n-1)/2. Số phép gán: trường hợp tốt nhất 2(n-1), xấu nhất la n(n+1)/2+1. Độ phức tạp của thuật toán O(n^2)
-

d) Bubble Sort

- Ý tưởng: xuất phát từ cuối(đầu) dãy đổi chỗ các cặp phần tử kế cận nhau để đưa phần tử nhỏ (lớn) hơn trong cặp phần tử đó về đầu (cuối) dãy và không xét đến nó ở những bước tiếp theo. Lặp lại quá trình cho đến khi không còn phần tử nào.
- Thuật toán:

Bước 1: i=0;
Bước 2: j=n-1, nếu j<i, xét a[j] và a[j-1], nếu a[j]<a[j-1] thì đổi chỗ a[j] và a[j-1], j=j-1;
Bước 3: i=i+1, nếu i<n-1 thì quay lại bước 2, ngược lại thì dừng.

- Đánh giá: số phép so sánh không phụ thuộc vào tình trạng của dãy ban đầu, số phép gán phụ thuộc vào số phép so sánh. Số phép so sánh: n(n-1)/2. Số phép gán: trường hợp tốt nhất là 0, xấu nhất là n(n-1)/2. Độ phức tạp của thuật toán là O(n^2)

e) Shaker Sort

- Ý tưởng: là một cải tiến của Bubble Sort, khi đưa phần tử nhỏ nhất về đầu mảng thuật toán sẽ đưa phần tử lớn nhất về cuối mảng.
- Thuật toán:

Bước 1: l=0,r=n-1;
Bước 2: i=r, khi i>l xét nếu a[i-1]>a[i] thì đổi chỗ a[i-1] và a[i], i=i-1. j=l+1, khi j<r xét nếu a[j+1]<a[j] thì đổi chỗ a[j+1] và a[j], j=j+1.
Bước 3: r=r-1, l=l+1, nếu l<r thì quay lại bước 2, ngược lại thì dừng.

- Đánh giá: so với Bubble Sort thì Shaker Sort có thời gian thực thi nhanh hơn. Độ phúc tạp của thuật toán tương đương với thuật toán Bubble Sort.

f) Shell Sort

- Ý tưởng: là một cải tiến của Insertion Sort, chia dãy cần sắp xếp thành một dãy con gồm các phần tử cách nhau h vị trí. Sắp xếp các phần tử trong dãy con, sau đó giảm khoảng cách h để tạo thành các dãy con mới và tiếp tục lặp lại việc sắp xếp trên. Thuật toán dừng khi h=1.
- Thuật toán:

Bước 1: chọn khoảng cách h
Bước 2: chia dãy ban đầu thành các dãy con cách nhau khoảng cách h. Sắp xếp dãy con bằng chèn trực tiếp
Bước 3: giảm khoảng cách h, nếu h=1 thì dừng thuật toán, ngược lại thì quay lại bước 2.

- Đánh giá: hiệu quả của thuật toán phụ thuộc vào dãy các độ dài được chọn

g) Quick Sort

- Ý tưởng: phân hoạch dãy ban đầu thành 2 phần, dãy 1: gồm các phần tử có giá trị nhỏ hơn x và dãy 2 gồm các phần tử có giá trị lớn x. Như vậy dãy ban đầu gồm 3 dãy(a[i]<x,a[i]=x và a[i]>x). Trong 2 dãy a[i]<x và a[i]>x ta xét nếu có 1 phần tử thì xem như đã có thứ tự, nếu lớn hơn 1 phần tử thì ta tiếp tục phân hoạch dãy con như dãy ban đầu.
- Thuật toán:

Bước 1: chọn một phần tử bất kỳ trong dãy a[i]=x làm giá trị mốc, i=l, j=r;
Bước 2: phân hoạch dãy đã cho thành 2 dãy với các phần tử a[i]>x và a[i]<x (a[i]<x thì i=i+1, a[j]>x thì j=j-1). Nếu a[i]>x>a[j] mà a[j] đứng sau a[i] thì dổi chỗ a[i] và a[j].
Bước 3: nếu i<j thì lặp lại bước 2, ngược lại dừng.

- Đánh giá: hiệu quả của giải thuật phụ thuộc vào việc chọn phần tử làm gốc. Trường hợp tốt nhất là khi chọn được phần tử gốc có giá trị giữa dãy. Trường hợp xấu nhất nếu chọn phần tử gốc là max hoặc min trong dãy. Độ phức tạp của giải thuật tong trường hợp tốt nhất là O(nlog(n)), trường hợp xấu nhất là O(n^2).

h) Merge Sort

- Ý tưởng: chia dãy thành 2 nữa sau đó kết hợp chúng lại với nhau để tạo thành mảng đã được sắp xếp. quá trình này được thực hiện đến khi không còn chia được nữa.
- Thuật toán:

Bước 1: l=0,r=n-1;

Bước 2: nếu l<r, tìm t=(r+l)/2 là chỉ số nằm giữa của mảng. Gọi đệ quy MergeSort cho nữa đầu tiên của dãy MergeSort(a,l,t). Gọi đệ quy hàm MergeSort cho nữa sau của dãy MergeSort(a,t+1,r);

Bước 3: ghép hai nữa dãy đã sắp xếp lại.

- Đánh giá: không tận dụng được những thông tin của dãy ban đầu nên trog mọi trường hợp chi phí của thuật toán là không đổi. Độ phức tạp của thuật toán O(nlog(n))

i) Heap Sort

- Ý tưởng: xây dựng một heap sao cho các nút cha có giá trị lớn hơn nút con. Khi đó nút gốc là nút có giá trị lớn nhất. Đổi chỗ nút gốc với nút lá cuối cùng (n-1), tiếp tục xây dựng heap mới với n-2 nút và đổi chỗ nút gốc với nút lá cuối cùng (n-2), tiếp tục thực hiện sau n-2 bước ta thu được dãy đã được sắp xếp.
- Thuật toán:

Bước 1: đưa phần tử lớn nhất về vị trí đứng ở cuối dãy, r=n, đổi chỗ a[1] và a[r];

Bước 2: loại bỏ phần tử lớn nhất ra khỏi heap, r=r-1. Chỉnh phần còn lại của dãy thành một heap mới;

Bước 3: nếu r>1 thì lặp lại bước 2, ngược lại dừng.

- Đánh giá: độ phức tạp của thuật toán O(nlog2(n))

j) Radix Sort

- Ý tưởng: sắp xếp theo cơ số. Giả sử mỗi phần tử trong dãy là một số nguyên có tối đa m chữ số. phân loại các phần tử này theo các chữ số hàng đơn vị, hàng chục, hàng trăm …
- Thuật toán:

Bước 1: k=0 là chỉ số phân loại hiện hành;

Bước 2: khởi tạo các mảng b[0], b[1] … b[9] rỗng;

Bước 3: cho i chạy từ 1 tới n. đặt a[i] vào lô b[t] với t là chữ số thứ k của a[i];

Bước 4: nối b[0], b[1], … b[9] lại;

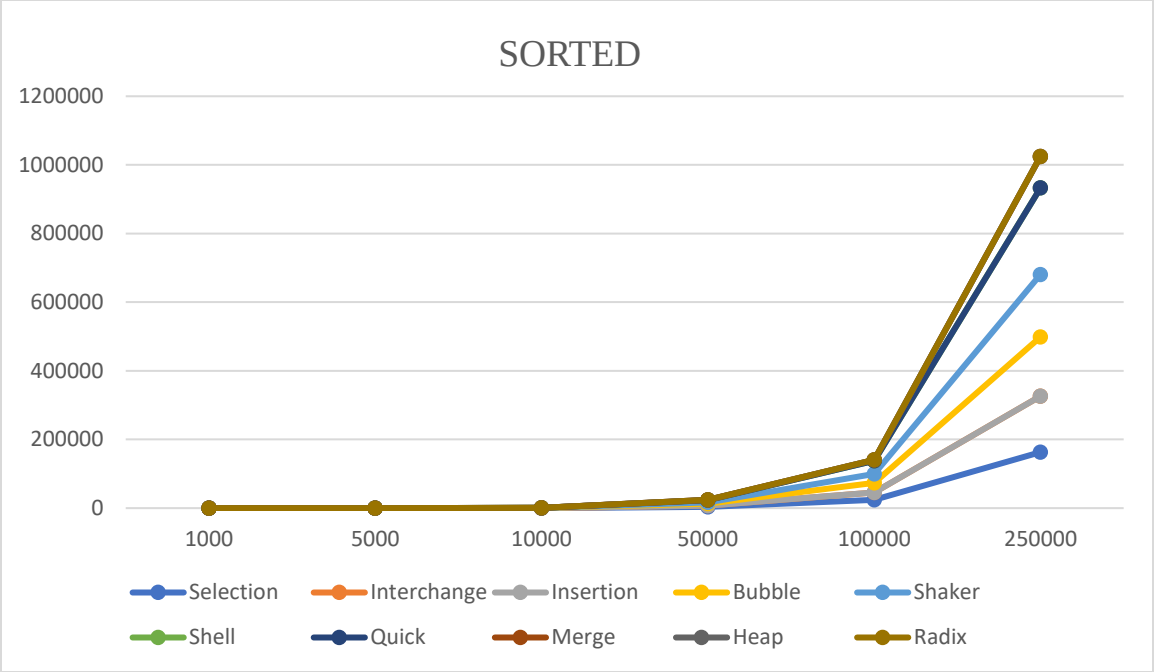
Bước 5: k=k+1, nếu k<m thì quay lại bước 2, ngược lại dừng.

- Đánh giá: độ phức tạp của thuật toán là O(n)

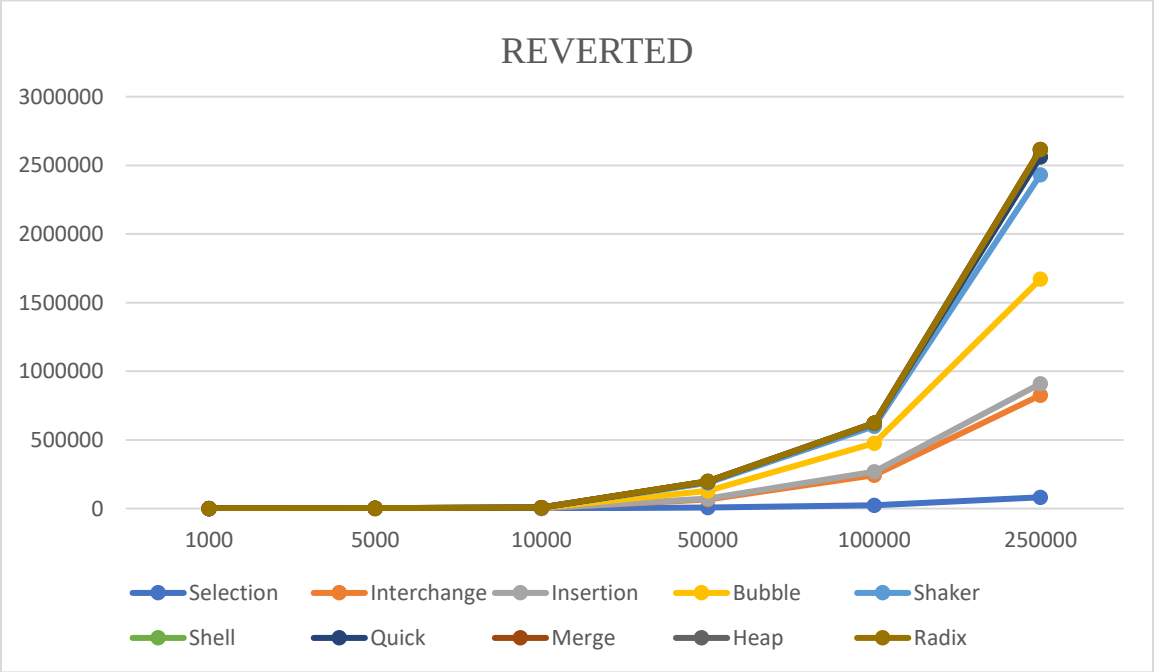
2. Kết quả thí nghiệm và nhận xét

a) Kết quả thí nghiệm

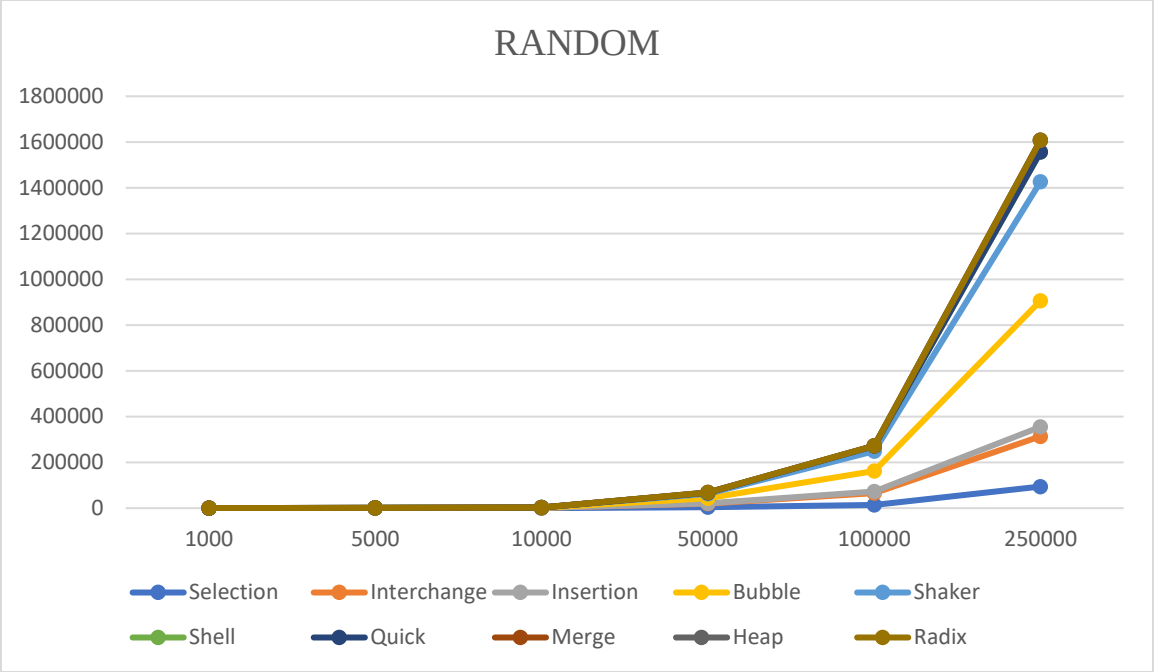
- Mảng đã có thứ tự Sorted:



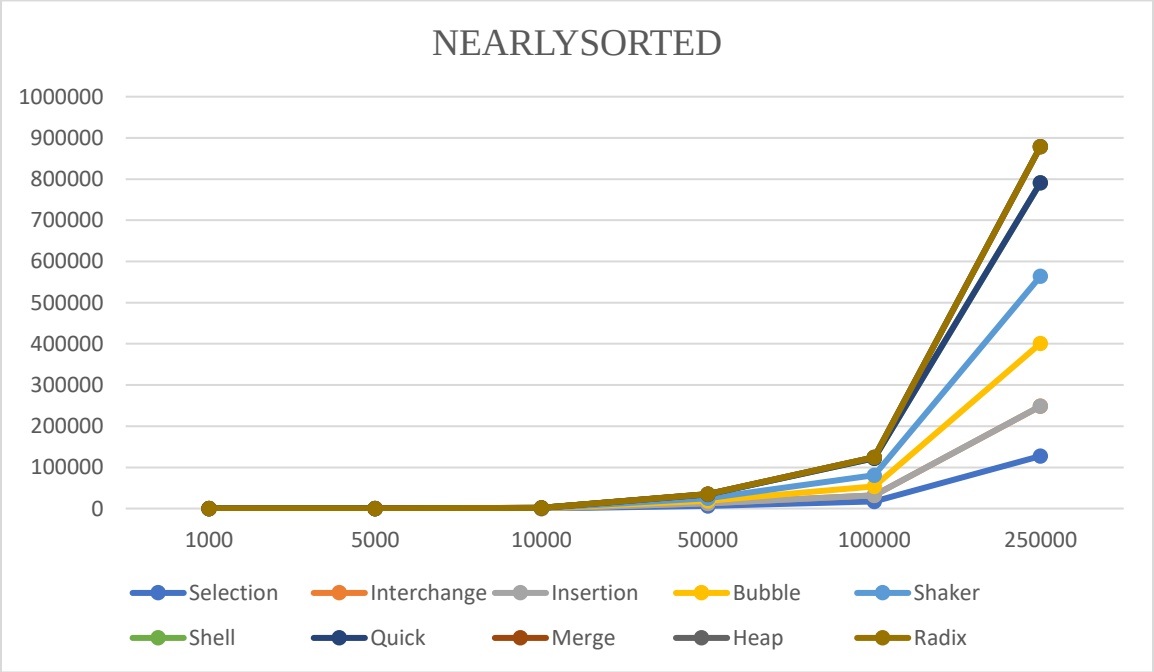
- Mảng có thứ tự ngược Reversed:



- Mảng có thứ tự ngâũ nhiên Random:



- Mảng gần như có thứ tự NearlySorted:



b) Nhận xét:

- Các thuật toán Selection Sort, Interchange Sort và Bubble Sort cài đặt đơn giản nhưng chi phí cao, thời gian thực thi lớn.
- Thuật toán Insertion Sort hiệu quả đối với các mảng dữ liệu đầu vào hầu như đã được sắp xếp.

- Thuật toán Merge Sort không tận dụng được tình trạng của dữ liệu mảng ban đầu, việc cài đặt khá phức tạp nhưng chi phí thấp.
- Shaker Sort được cải tiến từ Bubble Sort, chi phí cài đặt tương tự như thuật toán Bubble Sort, hiệu quả cải tiến vẫn chưa cao.
- Shell Sort hiệu phụ thuộc vào các khoảng được chọn.
- Thuật toán Heap Sort chi phí cài đặt thấp, thời gian thực thi nhanh nhưng việc cài đặt lại phức tạp.
- Quick Sort cài đặt phức tạp nhưng chi phí thấp, thời gian chạy nhanh hơn hẳn các thuật toán khác.
- Radix Sort tốc độ thực thi nhanh, hiệu quả đối với các mảng có số lượng phần tử lớn. thích hợp sắp xếp trên danh sách liên kết đơn.
- Như vậy mỗi thuật toán đều có ưu và nhược điểm riêng, ta có thể tùy vào số lượng phần tử của dãy, tình trạng dãy ban đầu mà chọn thuật toán sắp xếp phù hợp. Như khi số phần tử của dãy không nhiều ta có thể sử dụng các thuật toán dễ cài đặt như Selection Sort, Interchange Sort, đối vói dãy gần như có thứ tự nên sử dụng Insertion Sort bởi cài đạt khá đơn giản và hiệu quả cao đối với dãy đầu vào. Với số lượng phần tử của dãy lớn ta có thể sử dụng Quick Sort thời gian thực thi nhanh nhất. Còn đối với danh sách liên kết nên sử dụng Radix Sort.