Phần Báo Cáo

Tên : Lê Bá Quốc

MSSV : 18126008

Lớp: CNTT – K18 VP

Mục 1.1:

|  |  |  |  |
| --- | --- | --- | --- |
| Tên Thuật toán | Ý tưởng | Thuật toán | Đánh giá thuật toán |
| 1. Selection Sort | Chọn phần tử nhỏ nhất trong n phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu tiên của dãy hiện hành. Sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ còn n-1 phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2. Lặp lại quá trình trên cho dãy hiện hành đến khi dãy hiện hành chỉ còn một phần tử. Dãy ban đầu có n phần tử, vậy tóm tắt ý tưởng thuật toán là thực hiện n-1 lượt việc đưa phần tử nhỏ nhất trong dãy hiện hành về vị trí đúng ở đầu dãy. | * Bước 1: i=1 * Bước 2: Tìm phần tử a[min] nhỏ nhất trong dãy hiện hành từ a[i] đến a[n] * Bước 3: Hoán vị a[min] và a[i] * Bước 4: Nếu i<=n-1 thì i=i+1; Lặp lại bước 2 * Ngược lại: Dừng. n-1 phần tử đã nằm đúng vị trí. | 8/10 |
| 1. Merge Sort | Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp. | mergeSort(arr[], l, r)  If r > l  1. Tìm chỉ số nằm giữa mảng để chia mảng thành 2 nửa:  middle m = (l+r)/2  2. Gọi đệ quy hàm mergeSort cho nửa đầu tiên:  mergeSort(arr, l, m)  3. Gọi đệ quy hàm mergeSort cho nửa thứ hai:  mergeSort(arr, m+1, r)  4. Gộp 2 nửa mảng đã sắp xếp ở (2) và (3):  merge(arr, l, m, r) | 9/10 |
| 1. Heap sort | Sắp xếp vun đống (Heap Sort) là một kỹ thuật sắp xếp phân loại dựa trên một cấu trúc dữ liệu được gọi là đống nhị phân (binary heap), gọi đơn giản là đống. Nó tương tự như thuật toán [Sắp xếp chọn (Selection Sort)](https://dnmtechs.com/thuat-toan-sap-xep-chon-selection-sort/) nơi phần tử lớn nhất sẽ được xếp vào cuối danh sách. Lặp đi lặp lại các bước này cho các phần tử còn lại của danh sách. | Giai đoạn 1  Từ dãy dữ liệu input, ta sẽ sắp xếp chúng thành một heap (dạng cấu trúc cây nhị phân). Heap này có thể là Min-heap (nút gốc có giá trị bé nhất) hoặc Max-heap (nút gốc có giá trị lớn nhất), trong bài viết này, ta sẽ sử dụng Max-heap với một số yêu cầu thỏa mãn sau:   * Nút cha sẽ luôn lớn hơn tất cả các nút con, nút gốc của heap sẽ là phần tử lớn nhất. * Heap được tạo thành phải là một cây nhị phân đầy đủ, tức ngoại trừ các nút lá, ở cùng một cấp độ các nút nhánh không được thiếu.  Giai đoạn 2 Giai đoạn này gồm các thao tác được lặp đi lặp lại cho đến khi mảng dữ liệu được toàn tất sắp xếp:   1. Đưa phần tử lớn nhất của heap được tạo vào mảng kết quả, mảng này sẽ chứa các phần tử đã được sắp xếp. 2. Sắp xếp lại heap sau khi loại bỏ nút gốc (có giá trị lớn nhất) để tìm phần tử có giá trị lớn nhất tiếp theo. 3. Thực hiện lại thao tác 1 cho đến khi các phần tử của heap đều được đưa vào mảng kết quả.   Như thế, mảng kết quả sẽ chứa các phần tử được sắp xếp giảm dần. | 9/10 |
| 1. Quick Sort | 1. Chọn phần tử chốt. 2. Khai báo 2 biến con trỏ để trỏ để duyệt 2 phía của phần tử chốt. 3. Biến bên trái trỏ đến từng phần tử mảng con bên trái của phần tử chốt. 4. Biến bên phải trỏ đến từng phần tử mảng con bên phải của phần tử chốt. 5. Khi biến bên trái nhỏ hơn phần tử chốt thì di chuyển sang phải. 6. Khi biến bên phải nhỏ hơn phần tử chốt thì di chuyển sang trái. 7. Nếu không xảy ra trưởng hợp 5 và 6 thì tráo đổi giá trị 2 biến trái và phải. 8. Nếu trái lơn hơn phải thì đây là giá trị chốt mới. | Kỹ thuật chọn phần tử chốt :  Chọn phần tử đứng đầu hoặc đứng cuối làm hoặc đứng giữa phần tử chốt.  Chọn một yếu tố, được gọi là trục, từ danh sách.  Sắp xếp lại danh sách sao cho tất cả các phần tử có giá trị nhỏ hơn chốt xuất hiện trước chốt, trong khi tất cả các phần tử có giá trị lớn hơn chốt xuất hiện sau nó (giá trị bằng nhau có thể đi theo một trong hai cách). Sau khi phân vùng này, trục nằm ở vị trí hợp lý của nó. Đây được gọi là hoạt động phân vùng.  Hai thông tin chi tiết về phân vùng:  Tuyến tính (O (n)) trong thời gian, không có bộ nhớ thêm kể từ khi chúng tôi sử dụng giao dịch hoán đổi.  Giảm kích thước vấn đề - phân chia và chinh phục.  Sắp xếp đệ quy danh sách phụ của các phần tử nhỏ hơn và danh sách phụ của các phần tử lớn hơn. | 9/10 |
| 1. Bubble Sort | **Sắp xếp nổi bọt** ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh" \o "Tiếng Anh): *bubble sort*) là một [thuật toán sắp xếp](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_s%E1%BA%AFp_x%E1%BA%BFp" \o "Thuật toán sắp xếp) đơn giản, với thao tác cơ bản là so sánh hai phần tử **kề** nhau, nếu chúng chưa đứng đúng thứ tự thì đổi chỗ (*swap*). Có thể tiến hành từ trên xuống (bên trái sang) hoặc từ dưới lên (bên phải sang). | Bước 1: i=0; //Phần tử đầu tiên  Bước 2:Lần lượt so sánh và đổi chổ (nếu cần) từ phải sang trái đối với các phần từ từ a[n] đến a[i]. với biến gán j=n-i. và lặp lại khi j>i.  Bước 3: i=i+1  Bước 4:  Nếu i < n, quay lại Bước 2.  Ngược lại, dừng, dãy đã cho đã sắp xếp đúng vị trí. | 8/10 |
| 1. Insertion Sort | Thuật toán sắp xếp chèn thực hiện sắp xếp dãy số theo cách duyệt từng phần tử và chèn từng phần tử đó vào đúng vị trí trong mảng con(dãy số từ đầu đến phần tử phía trước nó) đã sắp xếp sao cho dãy số trong mảng sắp đã xếp đó vẫn đảm bảo tính chất của một dãy số tăng dần. | 1. Khởi tạo mảng với dãy con đã sắp xếp có k = 1 phần tử(phần tử đầu tiên, phần tử có chỉ số 0) 2. Duyệt từng phần tử từ phần tử thứ 2, tại mỗi lần duyệt phần tử ở chỉ số i thì đặt phần tử đó vào một vị trí nào đó trong đoạn từ [0…i] sao cho dãy số từ [0…i] vẫn đảm bảo tính chất dãy số tăng dần. Sau mỗi lần duyệt, số phần tử đã được sắp xếp k trong mảng tăng thêm 1 phần tử. 3. Lặp cho tới khi duyệt hết tất cả các phần tử của mảng. | 8/10 |
| 1. Radix Sort | Để sắp xếp dãy **a1, a2, …, an** giải thuật R**adix Sort** thực hiện như sau:   1. Trước tiên, ta có thể giả sử mỗi phần tử ai trong dãy a1, a2, …, an là một số nguyên có tối đa m chữ số. 2. Ta phân loại các phần tử lần lượt theo các chữ số hàng đơn vị, hàng chục, hàng trăm, . tương tự việc phân loại thư theo tỉnh thành, quận huyện, phường xã, .. | * Bước 1 : k cho biết chữ số dùng để phân loại hiện hành   k = 0; (k = 0: hàng đơn vị; k = 1: hàng chục)   * Bước 2 : Tạo các lô chứa các loại phần tử khác nhau   Khởi tạo 10 lô **B0, B1, .. , B9** rỗng;   * Bước 3 :   For i = 1 .. n do  Ðặt ai vào lô Bt với t = chữ số thứ k của ai;   * Bước 4 :   Nối **B0, B1, .. , B9** lại (theo đúng trình tự) thành a.   * Bước 5 :   **k = k+1**;  Nếu**k < m** thì trở lại bước 2.  Ngược lại: Dừng | 8/10 |

A screenshot of a cell phone

Description automatically generatedA picture containing screenshot

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a map

Description automatically generated

Nhận xét :

Trường hợp random :

-Selection sort và bubble sort là 2 thuật toán chạy chậm nhất từ dữ liệu 1000 – 100 000 (nhìn chung thì bubble sort chậm hơn với selection sort)

- Quick sort, Radix sort, Merge và Heap sort là thuật toán có tốc độ chạy ổn định nhất .

- những thuật toán còn lại có tốc độ chênh nhau không đáng kể (từ 0 đến 0.15 giây)

Trường hợp sorted:

-Selection sort và bubble sort là 2 thuật toán chạy chậm nhất từ dữ liệu 1000 – 100 000 (nhìn chung thì bubble sort chậm hơn với selection sort)

- Quick sort, Radix sort, Merge và Heap sort là thuật toán có tốc độ chạy ổn định nhất .

- những thuật toán còn lại có tốc độ chênh nhau không đáng kể (từ 0 đến 0.15 giây)

2 trường hợp còn lại tương tự.

Giải thích:

Dựa vào độ phức tạp của các thuật toán : chúng ta có thể dễ dàng nhận thấy được sự khác nhau giữa thời gian thực hiện các thuật toán trên .

Selection sort chậm nhất vì các trường hợp best – average - worst đều là O(n^2).

A screenshot of a cell phone

Description automatically generated