BÁO CÁO KẾT QUẢ TÌM HIỂU

Mục lục:

1. **Thuật toán**
2. Khái niệm………………………………………………………….2
3. Khái niệm độ khó của thuật toán………………………………….2
4. **Các thuật toán**
5. Thuật toán sắp xếp kiểu chọn(Selection Sort)

1.1.Ý tưởng……………………………………………………………2

1.2.Ưu điểm……………………………………………………………3

1.3.Nhược điểm………………………………………………………..3

1.4.Cài đặt thuật toán………………………………………………….3

1. **Thuật toán nổi bọt (Bubble Sort)**

2.1.Ý tưởng…………………………………………………………..3

2.2.Ưu điểm…………………………………………………………..3

2.3.Nhược điểm……………………………………………………….3

2.4.Cài đặt thuật toán…………………………………………………..4

1. **Sắp xếp chèn (Insertion Sort)**

3.1.Ý tưởng……………………………………………………………4

3.2.Ưu điểm……………………………………………………………4

3.3.Nhược điểm………………………………………………………..4

3.4.Cài đặt thuật toán…………………………………………………..4

1. **Sắp xếp trộn (Merge Sort)**

**4**.1.Ý tưởng……………………………………………………………4

4.2.Ưu điểm……………………………………………………………5

4.3.Nhược điểm………………………………………………………..5

4.4.Cài đặt thuật toán………………………………………………….5

1. **Sắp xếp nhanh (Quick Sort)**

**5**.1.Ý tưởng……………………………………………………………5

5.2.Ưu điểm……………………………………………………………6

5.3.Nhược điểm………………………………………………………..6

5.4.Cài đặt thuật toán…………………………………………………..6

1. **Sắp xếp bằng Heap (Heap Sort)**

2.1.Ý tưởng…………………………………………………………..6

2.2.Ưu điểm…………………………………………………………..6

2.3.Nhược điểm………………………………………………………6

2.4.Cài đặt thuật toán…………………………………………………7

**III. So sánh các thuật toán**

1. **Thuật toán**
2. **Khái niệm:**

**-** Trong [toán học](https://vi.wikipedia.org/wiki/To%C3%A1n_h%E1%BB%8Dc" \o "Toán học) và [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh" \o "Khoa học máy tính), một **thuật toán**, còn gọi là **giải thuật**, là một [tập hợp hữu hạn](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_h%E1%BB%A3p" \l "L%E1%BB%B1c_l%C6%B0%E1%BB%A3ng_c%E1%BB%A7a_t%E1%BA%ADp_h%E1%BB%A3p_-_H%E1%BB%AFu_h%E1%BA%A1n_v%C3%A0_v%C3%B4_h%E1%BA%A1n" \o "Tập hợp) các hướng dẫn [được xác định rõ ràng](https://vi.wikipedia.org/w/index.php?title=%C4%90%C6%B0%E1%BB%A3c_x%C3%A1c_%C4%91%E1%BB%8Bnh_r%C3%B5&action=edit&redlink=1" \o "Được xác định rõ (trang không tồn tại)), có thể thực hiện được bằng máy tính, thường để giải quyết một lớp vấn đề hoặc để thực hiện một phép tính. Các thuật toán luôn [rõ ràng](https://vi.wikipedia.org/w/index.php?title=T%E1%BB%91i_ngh%C4%A9a&action=edit&redlink=1" \o "Tối nghĩa (trang không tồn tại)) và được sử dụng chỉ rõ việc thực hiện các [phép tính](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A9p_t%C3%ADnh&action=edit&redlink=1" \o "Phép tính (trang không tồn tại)), [xử lý dữ liệu](https://vi.wikipedia.org/wiki/X%E1%BB%AD_l%C3%AD_d%E1%BB%AF_li%E1%BB%87u" \o "Xử lí dữ liệu), [suy luận tự động](https://vi.wikipedia.org/wiki/Suy_lu%E1%BA%ADn_t%E1%BB%B1_%C4%91%E1%BB%99ng" \o "Suy luận tự động) và các tác vụ khác.

1. **Khái niệm độ phức tạp của thuật toán:**

-  Mỗi một bài toán có giới hạn/kích thước của đầu vào. Độ phức tạp thuật toán là 1 khái niệm/định nghĩa/định lượng tương đối thể hiện số phép toán của giải thuật so với kích thước của đầu vào.

      Thời gian thực hiện một giải thuật bằng chương trình máy tính phụ thuộc vào các yếu tố:

      • Kích thước dữ liệu đầu vào: một giải thuật hay một chương trình máy tính thực hiện trên tập dữ liệu có kích thước lớn hiển nhiên mất nhiểu thời gian hơn thuật toán hoặc chương trình này thực hiện trên tập dữ liệu đầu vào có kích thước nhỏ.

      • Phần cứng của hệ thống máy tính: hệ thống máy tính có tốc độ cao thực hiện

nhanh hơn trên hệ thống máy tính có tốc độ thấp.

     Gọi độ dài dữ liệu đầu vào là *T*(*n*). Khi đó, số lượng các phép toán sơ cấp để giải bài toán *P* thực hiện theo thuật toán *F*=*F*1*F*2..*F*n trên độ dài dữ liệu *T*(*n*) là *F*(*T*(*n*)). Để xác định số lượng các phép toán sơ cấp *F*i (*i*=*1*, *2*, .., *n*) thực hiện trong thuật toán *F* ta cần phải giải bài toán đếm để xác định *F*(*T*(*n*)). Đây là bài toán vô cùng khó và không phải lúc nào cũng giải được []. Để đơn giản điều này, người ta thường tìm đến các phương pháp xấp xỉ để tính toán độ phức tạp thời gian của một thuật toán. Điều này có nghĩa, khi ta không thể xây dựng được công thức đếm *F*(*T*(*n*)), nhưng ta lại có khẳng định chắc chắn *F*(*T*(*n*)) không vượt quá một phiếm hàm biết trước *G*(*n*) thì ta nói *F*(*T*(*n*)) thực hiện nhanh nhất là *G*(*n*).

1. **Các thuật toán**
2. **Thuật toán sắp xếp kiểu chọn(Selection Sort)**
   1. **Ý tưởng:**

- Ý tưởng thuật toán cũng giống như việc xếp hàng trong giờ thể dục.Thầy giáo thể dục muốn xếp các bạn trong lớp thành một hàng theo thứ tự từ thấp đến cao, thầy so sánh chiều cao của 2 bạn học sinh đứng cạnh nhau trong hàng, nếu bạn bên phải thấp hơn bạn bên trái thì đổi chỗ 2 bạn cho nhau.

* 1. **Ưu điểm:**

- Là thuật toán cơ bản, dễ hiểu, phù hợp cho người bắt đầu học về sắp xếp.

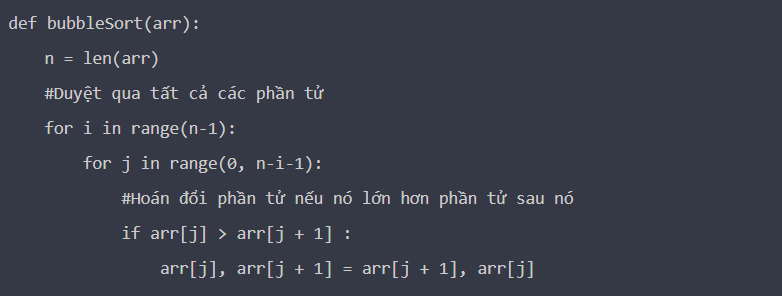
- Đoạn code ngắn gọn, dễ nhớ.

* 1. **Nhược điểm:**

- Hiệu xuất thấp nhất trong các thuật toán sắp xếp.

- Không hiệu quả với những dữ liệu lớn.

* 1. **Cài đặt ý thuật toán:**



*Hình 1. Thuật toán sắp xếp chọn*

1. **Thuật toán nổi bọt (Bubble Sort)**
   1. **Ý tưởng:**

**-** Là chúng ta sắp xếp các quân bài trên tay bằng cách là sẽ tìm ra quân bài nhỏ nhất trên tay và lần lướt đặt chúng xuống cho đến khi hết các quân bài trên tay.

* 1. **Ưu điểm:**

-Thuật toán dễ tìm hiểu dễ triển khai

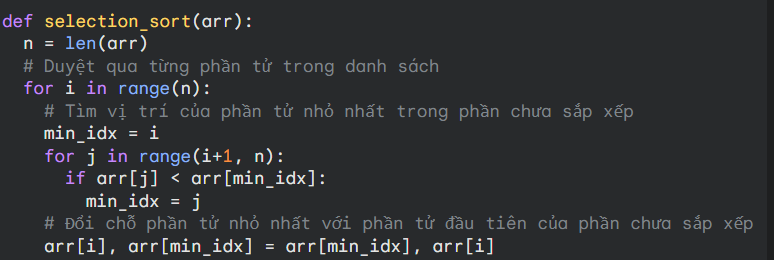
- **Ổn định:** Các phần tử có giá trị bằng nhau sẽ giữ nguyên thứ tự tương đối của chúng sau khi sắp xếp.

**2.3 Nhược điểm:**

-**Hiệu suất thấp:** Độ phức tạp thời gian trung bình và tệ nhất của thuật toán là O(n²), điều này khiến nó trở nên kém hiệu quả với các danh sách lớn.

-**Nhiều phép so sánh:** Thuật toán thực hiện một số lượng lớn phép so sánh, đặc biệt là với các danh sách lớn, dẫn đến thời gian thực thi lâu.

* 1. **Cài đặt thuật toán**:



*Hình 2. Thuật toán sắp xếp nổi bọt*

**3.Sắp xếp chèn (Insertion Sort)**

**3.1.Ý tưởng:**

- Giống khi ta sắp xếp các quân bài trên mặt bàn ta chọn lá đầu tiên làm mốc so sánh lá thứ 2 nếu nhỏ hơn thì sẽ chèn vào chỗ của lá đầu tiên và lần lượt với lá tiếp theo vào chèn chúng vào đúng vị trí của chúng cần được sắp xếp.

**3.2.Ưu điểm:**

- Đơn giản và dễ thực hiện.

- Thuật toán sắp xếp ổn định

- Hiệu quả cho danh sách nhỏ và danh sách gần như được sắp xếp.

**3.3.Nhược điểm:**

- Không hiệu quả cho các danh sách lớn.

- Không hiệu quả với hầu hết các thuật toán sắp xếp khác.

**3.4.Cài đặt thuật toán:**



*Hình 3. Thuật toán sắp xếp chèn*

**4.Sắp xếp trộn (Merge Sort)**

**4.1.Ý tưởng:**

- Là ta chia dãy ra thành các phần tử bằng nhau sau đó gộp 2 phần từ thành các dãy có kích thước bằng nhau và so sánh chúng rồi gộp cái phần tử bé hơn và lớn hơn lại với nhau sau đó sắp xếp và hợp lại dãy kịch thước ban đầu.

**4.2.Ưu điểm:**

- Có độ phức tạp ổn định.

- Phù hợp với những bài có dữ liệu lớn.

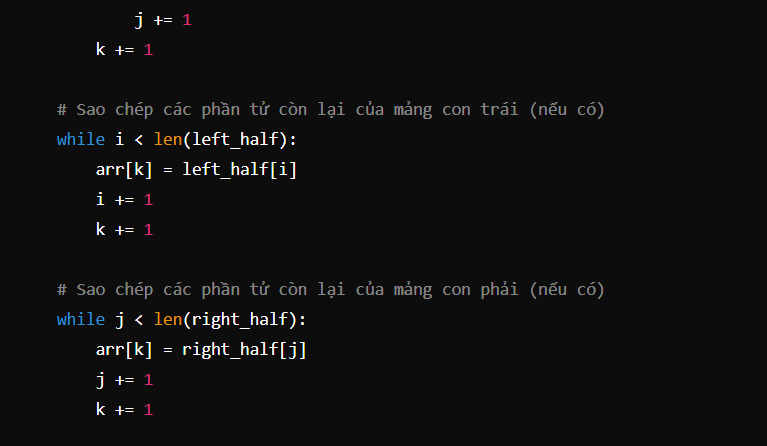
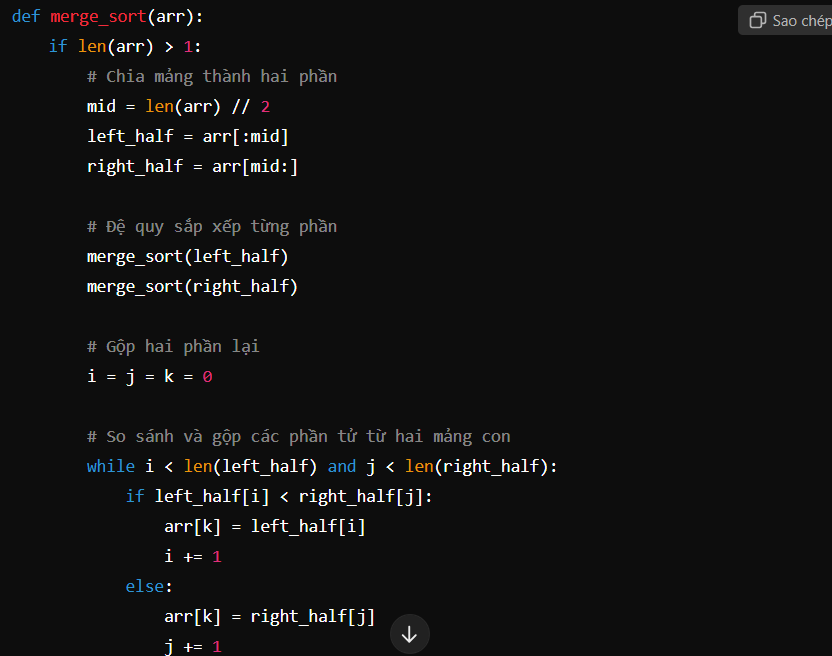
**4.3.Nhược điểm:**

- Tốn bộ nhớ

- Không hiệu quả với dữ liệu nhỏ

- Triển khai phức tạp hơn thuật toán khác

**4.4.Cài đặt thuật toán.**



*Hình 4.Thuật toán sắp xếp trộn*

1. **Sắp xếp nhanh (Quick Sort)**
   1. **Ý tưởng**

- Sắp xếp nhanh chọn một phần tử làm chốt (pivot), chia mảng thành hai phần: phần nhỏ hơn pivot và phần lớn hơn pivot, rồi sắp xếp đệ quy hai phần này.

* 1. **Ưu điểm:**

- Tốc độ sắp xếp nhanh.

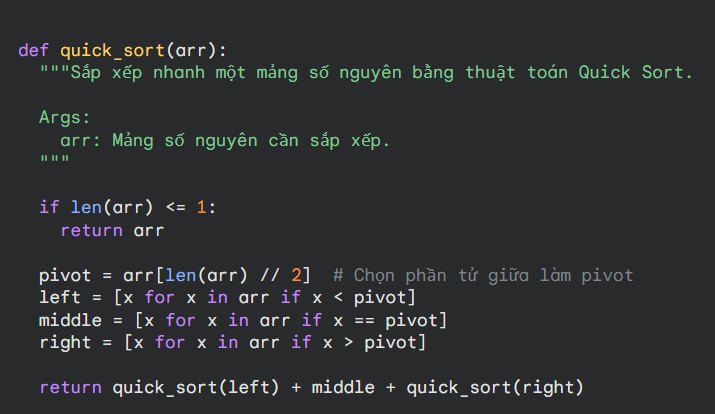
- Dùng được cho nhiều thuật toán

* 1. **Nhược điểm:**

- Phụ thuộc vào phần tử chốt.

- Không ổn định

* 1. **Cài đặt thuật toán**



*Hình 5.Thuật toán sắp xếp nhanh*

1. **Sắp xếp bằng Heap (Heap Sort)**
   1. **Ý tưởng:**

- Chia mảng ban đầu thành hai mảng con: Một mảng bao gồm các phần tử đã sắp xếp và một mảng bao gồm các phần tử còn lại chưa được sắp xếp. Ban đầu, mảng đã sắp xếp là mảng rỗng, mảng chưa sắp xếp chính là mảng ban đầu.

- Chia mảng ban đầu thành hai mảng con: Một mảng bao gồm các phần tử đã sắp xếp và một mảng bao gồm các phần tử còn lại chưa được sắp xếp. Ban đầu, mảng đã sắp xếp là mảng rỗng, mảng chưa sắp xếp chính là mảng ban đầu.

* 1. **Ưu điểm:**

- **Hiệu suất cao:** Heap Sort có độ phức tạp thời gian trung bình và tệ nhất là O(n log n), tương đương với Quick Sort.

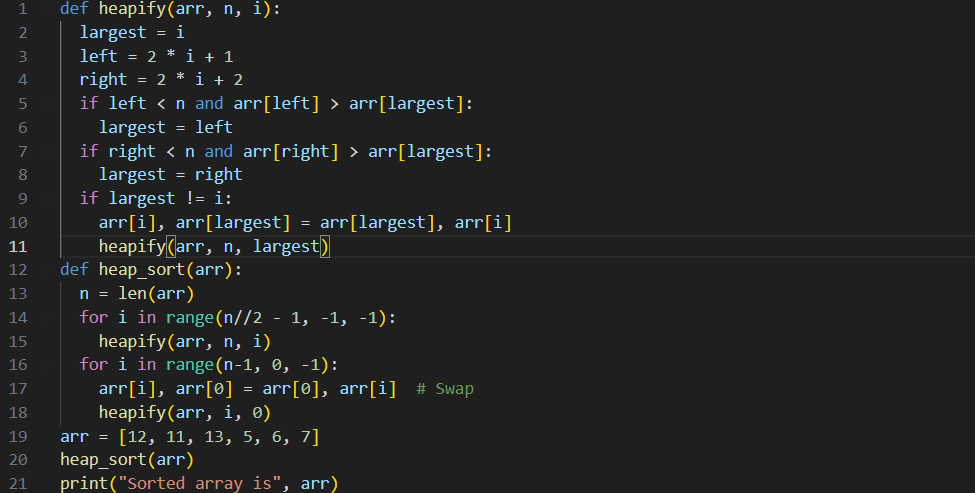
- Ổn định và dễ triển khai

* 1. **Nhược điểm:**

- Khá là khó hiểu hơn so với các thuật toán khác

- Không linh hoạt thường thì thuật toán sẽ thực hiện cho cả mảng

* 1. **Cài đặt thuật toán**



*Hình 6. Thuật toán sắp xếp heap*

1. **So sánh các thuật toán**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thuật toán | Ý tưởng | Ưu điểm | Nhược điểm | Độ phức tạp |
| Bubble sort | So sánh và hoán đổi | Dễ hiểu dễ cài đặt | Hiệu xuất thấp | O(n^2) |
| Selection sort | Tìm phần tử nhỏ nhất và hoán đổi | Dễ cài đặt | Không hiệu quả với tập dữ liệu lớn | O(n^2) |
| Insertion sort | Chèn phần tử vào vị trí đúng | Hiệu quả với mảng nhỏ hoặc gần được sắp xếp | Không hiệu quả cho tập dữ liệu lớn | O(n^2) |
| Merge sort | Chia và kết hợp | Hiệu quả cho tập dữ liệu lớn | Cần thêm bộ nhớ phụ | O(n log n) |
| Quick sort | Chọn trụ và phân chia | Hiệu quả, nhanh với tập dữ liệu lớn | Hiệu xuất giảm với mảng đã sắp xếp | O(n log n) trung bình |
| Heap sort | Xây dựng heap và trích xuất | Ổn định, hiệu quả | Cần hiểu biết về heap | O(n log n) |
| Counting sort | Đếm và sắp xếp theo số lần xuất hiện | Nhanh với các mảng nhỏ và tệp giá trị nhỏ | Không hiệu quả với dải giá trị lớn | O( n+k ) |