Thuật toán sắp xếp (P1)

Hello, chào mừng tất cả các bạn đã đến với một trong chuỗi bài ôn tập "Study with me" của Ban học tập Đoàn khoa Công nghệ phần mềm ^^ Hôm nay chúng mình sẽ cùng các bạn đến với môn ***Cấu trúc dữ liệu và Giải thuật***, cụ thể là phần giải thuật sắp xếp và một số bài tập. Nào, let's goooooooo!

Trước khi bắt đầu đến với các giải thuật sắp xếp, các bạn có đặt ra câu hỏi: ***Tại sao cần phải sắp xếp?***

Thế cho nên, điều đầu tiên chúng mình muốn đem đến cho các bạn đó là câu trả lời cho câu hỏi này.

Hãy thử tưởng tượng bạn cần tra cứu một từ trong từ điển, tuy nhiên cuốn từ điển đó lại không được sắp xếp theo thứ tự alphabet, các từ trong từ điển được sắp xếp theo một quy luật hoàn toàn ngẫu nhiên.

- Khi đó, việc bạn phải làm là **lật từng trang**, chưa hết đâu, cứ mỗi trang như vậy, bạn lại còn phải dò từng từ xem từ bạn cần có trong trang đó hay không.
- Và việc này khiến bạn phải bỏ ra khá nhiều thời gian và công sức. Tuy nhiên, nó sẽ "i-zi" hơn nhiều khi bạn có một cuốn từ điển đã được sắp xếp sẵn theo thứ tự alphabet.

Hay thử lấy một ví dụ khác, nếu như bạn được cho một danh sách điểm của sinh viên toàn trường, và yêu cầu tìm ra sinh viên có số điểm cao nhất.

- Ok, công việc này vẫn khá dễ dàng nhỉ vì chỉ cần duyệt qua danh sách một lần và tìm ra sinh viên có điểm cao nhất là được.
- Nhưng, sẽ như thế nào nếu bạn được yêu cầu tìm ra **top 10 sinh viên có số điểm cao nhất?** Nghĩ thôi cũng thấy quá tốn thời gian và công sức, chắc chỉ có tổ tiên mách bảo thì mới làm nhanh được thôi :>
- Tuy nhiên, nếu bạn có một danh sách đã được sắp xếp sẵn số điểm theo thứ tự giảm dần, thì việc bạn cần làm, có phải là chỉ cần lấy ra 10 sinh viên đầu tiên trong danh sách thôi đúng không?
- Đúng vậy, **mọi thứ đã trở nên dễ dàng hơn rất nhiều rồi.**

Qua những ví dụ kể trên, các bạn có thể nhận thấy:

- ✓ Sắp xếp là việc làm cho (xử lý) một danh sách các phần tử (hoặc các mẫu tin) để đặt chúng theo một thứ tự thỏa mãn một tiêu chuẩn nào đó dựa trên nội dung thông tin lưu giữ tại mỗi phần tử.
- ✓ Sắp xếp giúp cho việc tìm kiếm dễ dàng, hiệu quả và nhanh chóng.

Wow, tiện dụng như thế ^^ dại gì mà không sắp xếp nhỉ? =))

Phần 1: Những thuật toán thông dụng:

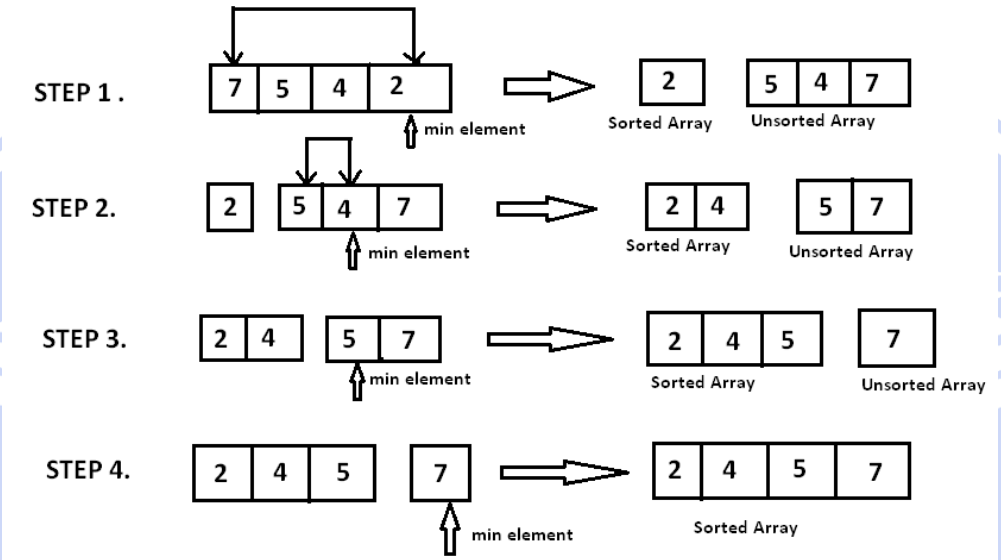
1. Sắp xếp chọn trực tiếp (Selection sort)

Đến với chiếc thuật toán đầu tiên là sắp xếp chọn :DDD

Thuật toán sắp xếp chọn trực tiếp giống như việc tìm người thích hợp và hoán đổi.

Hãy thử tưởng tượng như nếu bạn đang ở chức vụ trưởng phòng nhưng trình độ của bạn lại thấp hợp những người cấp dưới. Thì có phải cấp trên của bạn phải tìm kiếm những người làm chức vụ thấp hơn bạn nhưng trình độ cao và thích hợp hơn, sau đó hoán đổi chức vụ của bạn với người thích hợp đúng không?

Để dễ hiểu mình có 1 bức ảnh minh họa cho các bạn (trình độ càng cao, giá trị càng nhỏ):



Như bạn thấy kể từ người có "Mức trình độ 7" thì so với người có "Mức trình độ 2" thì phải hoán đổi hai người này đúng không? Và tương tự từ người có "Mức trình độ 5" thì so với người có "Mức trình độ 4" thì cũng phải cần giáng chức anh 5 và thăng chức anh 4 đúng không nào. Và cứ thế mà làm thôi cho đến khi nào hệ thống nhân viên công ti đạt chuẩn thì mình dừng.

Thuật toán này các bạn thường áp dụng trong các trường hợp trong đời sống, ví dụ như bầu lớp trưởng, thì phải xem những bạn dưới lớp trưởng có còn ai thích hợp hơn không để chúng ta đề cử và blah blah blah.

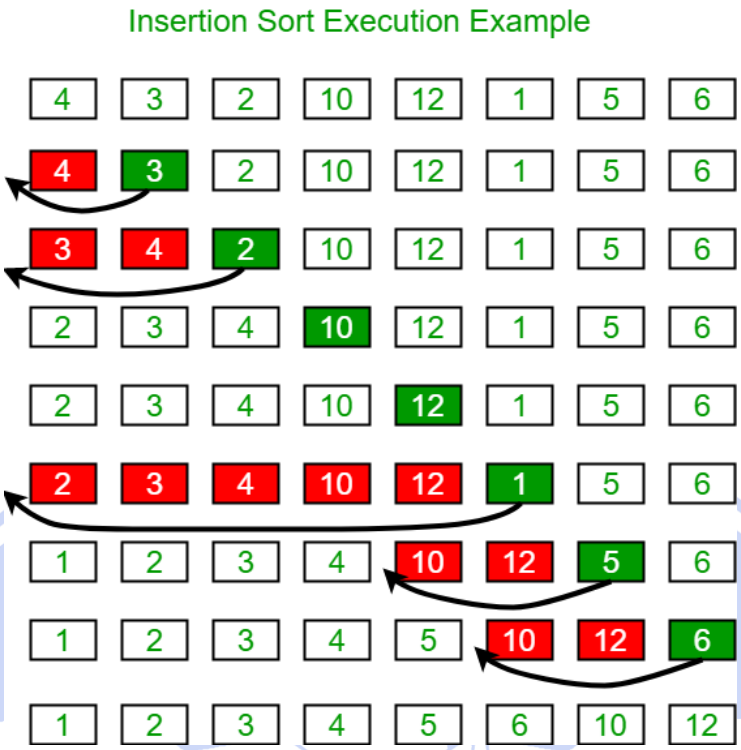
Link minh họa thuật toán Selection sort: https://www.youtube.com/watch?v=Ns4TPTC8whw

2. Sắp xếp chèn trực tiếp (Insertion Sort)

Bây giờ chúng ta sẽ chuyển qua một thuật toán, khá là gần gũi với đời sống, đặc biệt là các dịp lễ, tết =))

Thuật toán sắp xếp chèn, được mô phỏng từ cách mà chúng ta sắp xếp những quân bài khi chơi. Ok, vậy cùng nhau nhớ lại chúng ta sắp xếp bài được chia ra như thế nào nhỉ? Để sắp một bộ bài theo trật tự thì chúng ta rút lần lượt từ quân thứ 2, so với các quân đứng trước nó để chèn vào vị trí thích hợp.

Đến đây chắc mọi người đã mường tượng ra ý tưởng của giải thuật rồi đúng không nè :> vậy thì hãy cùng chúng mình xem qua một ví dụ để hiểu rõ hơn nha :>



Có thể thấy, thuật toán sắp xếp chèn về cơ bản có cách thức hoạt động khá giống với thuật toán sắp xếp nổi bọt (*Bubble sort, sẽ được giới thiệu ở phần 2*). Tuy nhiên, hãy đánh giá qua một lượt về thuật toán này nhé:

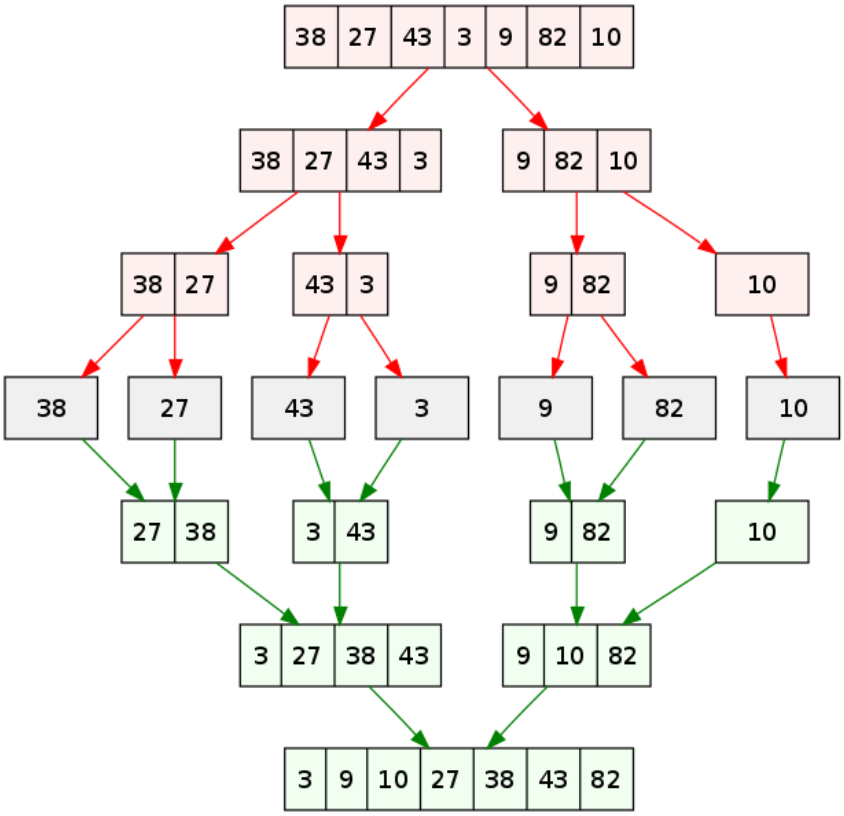
- Cũng tương tự như sắp xếp chọn, thuật toán sắp xếp chèn cũng có độ phức tạp thời gian trung bình là $O(n^2)$ do có hai vòng lặp lồng vào nhau.
- Thuật toán thích hợp đối với mảng đã được sắp xếp một phần hoặc mảng có kích thước nhỏ.

- Link minh họa thuật toán Insert-sort: https://www.youtube.com/watch?v=ROalU379l3U

3. Sắp xếp trộn (Merge sort)

Nào, bây giờ chúng ta cùng chuyển qua một thuật toán mới, với nguyên tắc hoạt động dựa trên một câu slogan mà chúng ta có thể đã gặp qua nhiều lần ở môn Lịch sử :< Đúng vậy, bạn đang nghĩ đúng đấy, nó chính là **"chia để trị"**

Và đây là cách thức thuật toán sắp xếp trộn hoạt động, cùng xem qua một ví dụ nhé! ^^



Đây là toàn bộ sơ đồ tiến trình của thuật toán merge sort cho mảng **{38, 27, 43, 3, 9, 82, 10}**. Nếu nhìn kỹ hơn vào sơ đồ này, chúng ta có thể thấy mảng ban đầu được lặp lại hành động chia cho tới khi kích thước các mảng sau khi chia là 1. Khi kích thước mảng con là 1, tiến trình gộp sẻ bắt đầu thực hiện giộ lại các mảng này cho tới khi hoàn thành và chỉ còn 1 mảng đã sắp xếp.

Với trường hợp khi 2 mảng con chỉ có 1 phần tử, ta chỉ xem phần tử nào nhỏ hơn và đẩy lên đầu, phần tử còn lại đặt phí sau. Do vậy, các mảng con cần gộp lại có tính chất luôn được sắp tang dần.

Merge sort có khả năng sắp xếp nhanh hơn so với một số thuật toán cơ bản như Insertion sort, Selection sort, Interchange sort.

Tuy nhiên, cái gì cũng có hai mặt cả, bên cạnh ưu điểm thì thuật toán sắp xếp chèn cũng có trong mình một số nhược điểm:

- Thuật toán khó cài đặt, không nhận dạng được mảng đã được xếp.
- Hiệu quả được tính bằng công thức O(n log n).

- Link minh họa thuật toán Merge sort: https://bitly.com.vn/1eddbr

4. Sắp xếp vun đống (Heap sort)

Đến với phần tiếp theo chúng ta sẽ làm quen với em Heap sort (đọc tên thôi thấy cũng sợ nhưng cũng không đáng sợ lắm đâu).

Để cho dễ hiểu thì chúng ta sẽ minh họa lại dãy thành dạng cây và liên tục hiệu chỉnh lan truyền, nghe lạ lắm phải không??

Share nhẹ cái hình cho coi thử nè:

- Ảnh động:
- Minh họa thuật toán Heap sort:

Nhìn vào một cái cây thì các bạn ắt hẳn biết được phần nào là đỉnh của nó chứ? Đỉnh là cái đầu nhọn còn 2 cái nằm 2 bên ngang hàng nhau nối vào 1 đỉnh được coi là lá của đỉnh.

Vậy dựa theo ảnh động ta thấy cứ mỗi lần cây mọc ra một cái lá thì nó luôn phải so sánh với cái đỉnh của nó. Nếu nó mà lá lớn hơn đỉnh thì mình đổi chỗ lá và đỉnh và mình tiếp tục như vậy cho đến khi nó không thỏa hoặc là đã đến đỉnh trên cùng.

Bước dời lá lên đỉnh như vậy chúng ta gọi nó là "Hiệu chỉnh lan truyền".

Bước hiệu chỉnh lan truyền còn 1 bước nữa là phải so sánh các lá để lựa chọn xem lá nào là lá nhỏ lớn nhất trong 2 lá để mình bắt đầu hiệu chỉnh.

Sau mỗi lần hiệu chỉnh như vậy các bạn có thể thấy tại mỗi đỉnh so với các lá con tại đỉnh đó thì nó lớn hơn đúng không? Và người ta gọi nó là "Cây cực đại" (Max heap)

Vậy là các bạn đã nắm được các thuật ngữ mà khi thầy cô giảng về Heap sort cho mình nhưng nghe cứ lớ ngớ.

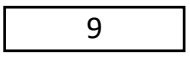
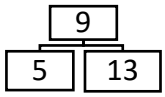
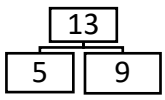
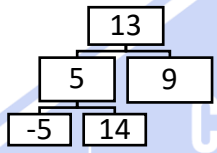
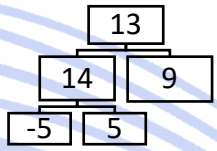
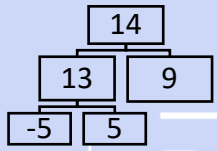
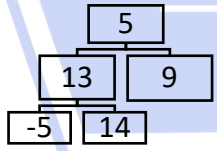
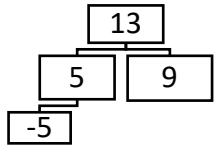
- Nhưng mới hiệu chỉnh thôi mà, chưa có sắp xếp gì cả. Vận dụng đặc điểm của cây cực đại ta thấy đỉnh trên cùng (Gốc đỉnh) là phần tử lớn nhất ➔Vậy ta chỉ việc lấy đi phần tử đó đặt vào vị trí thích hợp thôi. Tùy vào sắp xếp tăng dần hay giảm dần mà mình đặt vào cuối hay đầu danh sách.
- Nhưng lấy đi cái đỉnh rồi thì cây nó bị gãy làm sao??? Nên là chúng ta không thể lấy một cách tự nhiên như thế được. Theo ảnh động các bạn có thể thấy là nó hoán đổi cái đỉnh với cái lá cuối cùng trong đỉnh cuối cùng rồi mới bắt đầu ngắt đi cái lá cực đại đó đi đúng không?
- Sau đổi chỗ và ngắt lá như vậy thì cái cây mình lúc này bị thay đổi và không còn là cây cực đại nên là theo ảnh động thì chúng ta lại tiếp tục hiệu chỉnh lan truyền cho đến đỉnh cuối cùng đúng không? Lặp lại các bước:

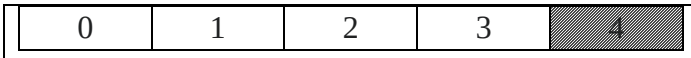
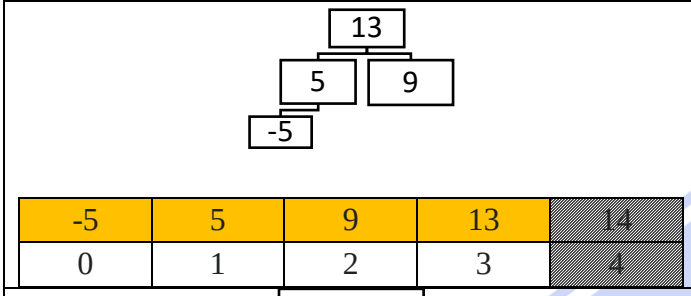
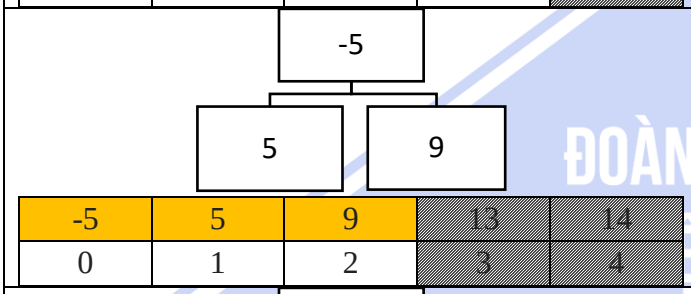
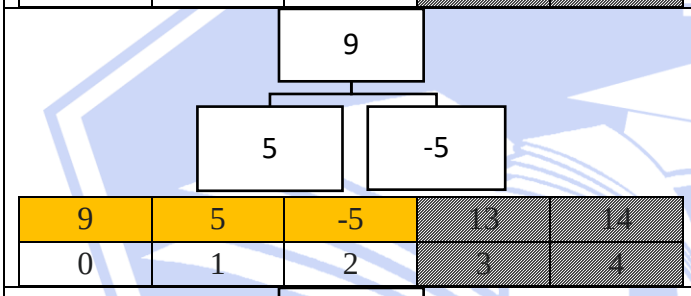
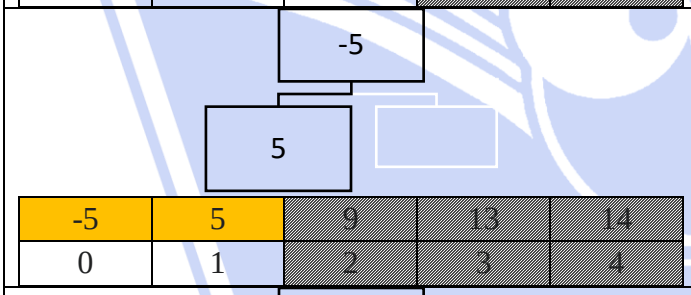
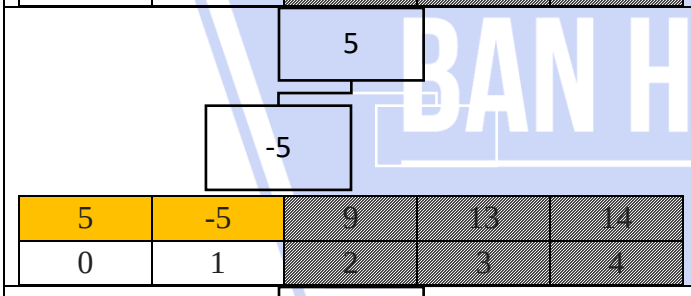
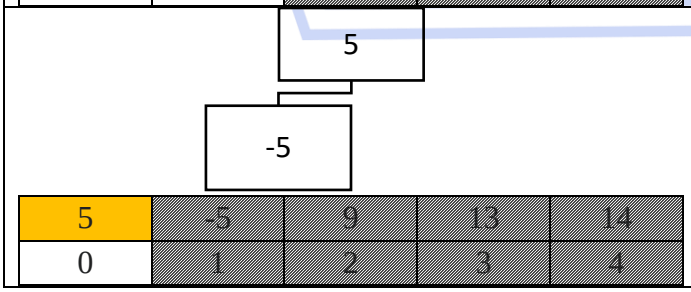
Hiệu chỉnh từ đỉnh đầu đến đỉnh cuối → Hoán đổi gốc - đỉnh → Ngắt lá.

Cứ như vậy cho đến khi cây còn 1 phần tử là kết thúc.

Ở đây mình ví dụ 1 cái mảng chỉ 5 phần phần tử thôi:

9	5	13	-5	14

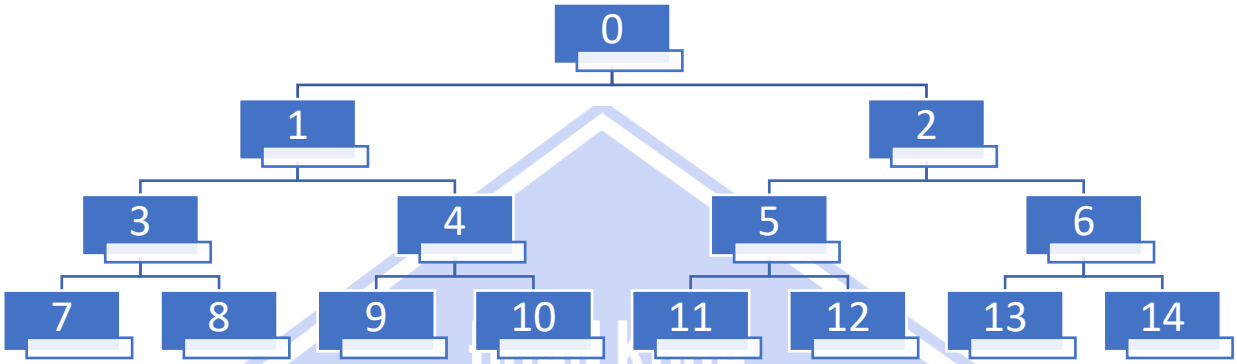
 9 | 5 | 13 | -5 | 14 0 | 1 | 2 | 3 | 4	
 9 | 5 | 13 | -5 | 14 0 | 1 | 2 | 3 | 4	Xét 5 và 13 thấy lá 13 lớn hơn và thấy lá 13 lớn hơn đỉnh 9 ➔ Đổi chỗ lá 13 với đỉnh 9
 13 | 5 | 9 | -5 | 14 0 | 1 | 2 | 3 | 4	Sau khi hiệu chỉnh thì đã ở đỉnh gốc nên dừng hiệu chỉnh
 13 | 5 | 9 | -5 | 14 0 | 1 | 2 | 3 | 4	Xét -5 và 14 thấy lá 14 lớn hơn và thấy lá 14 lớn hơn đỉnh 5 ➔ Đổi chỗ lá 14 với đỉnh 5
 13 | 14 | 9 | -5 | 5 0 | 1 | 2 | 3 | 4	Tiếp tục hiệu chỉnh vì chưa đến đỉnh gốc Xét 14 và 9 thấy lá 14 lớn hơn và thấy lá 14 lớn hơn đỉnh 13 ➔ Đổi chỗ lá 14 với đỉnh 13
 14 | 13 | 9 | -5 | 5 0 | 1 | 2 | 3 | 4	Lúc này đã đến đỉnh gốc nên dừng hiệu chỉnh Lúc này đã hiệu chỉnh đỉnh cuối cùng nên bắt đầu hoán đổi đỉnh với chiếc lá cuối cùng (Đầu mảng với phần tử cuối mảng đang xét)
 5 | 13 | 9 | -5 | 14 0 | 1 | 2 | 3 | 4	Lúc này chúng ta thu lại mảng đang xét 1 phần tử và tiếp tục hiệu chỉnh từ đỉnh gốc đến đỉnh cuối cùng và lại hoán vị phần đỉnh gốc với chiếc lá cuối cùng. Tiếp tục thu lại mảng và hiệu chỉnh cho đến khi mảng được thu lại chỉ còn 1 phần tử.
 13 | 5 | 9 | -5 | 14	Hoán đổi 13 với 5

	
	Hoán đổi đỉnh gốc và lá cuối cùng và thu hẹp mảng
	Hoán đổi 9 với -5
	Hoán đổi đỉnh gốc và lá cuối cùng và thu hẹp mảng
	Hoán đổi 5 với -5
	Hoán đổi đỉnh gốc và lá cuối cùng và thu hẹp mảng
	Nhận lại mảng chỉ còn 1 phần tử nên kết thúc tại đây.

Mảng nhận lại được sau sắp xếp:

5	-5	9	13	14

Nhưng đó là minh họa trên cây, vậy trên mảng thì làm sao mình biết cái nào là đỉnh, cái nào là lá đúng chứ? Các bạn tìm ra được qui luật sau đây:



Như các bạn thấy cứ mỗi đỉnh ta có 2 chiếc lá và 2 chiếc lá nó được tính lần dựa theo đỉnh kề của nó bằng công thức 2*i+1 và 2*i+2 (với i là đỉnh chứa lá)

VD:

- Với đỉnh 0 mình có 2 chiếc lá *2*0+1* và *2*0+2* là **1** và **2**.
- Với đỉnh 0 mình có 2 chiếc lá *2*1+1* và *2*1+2* là **3** và **4**.
- Với đỉnh 5 mình có 2 chiếc lá *2*5+1* và *2*5+2* là **11** và **12**.

⇨ Như vậy là bạn có thể chuyển từ cây sang dãy (mảng) rồi. Và dựa công thức chuyển đổi cũng như minh họa thuật toán là bạn có thể tự tay mình build những dòng code cho thuật toán Heap sort rồi. Sướng chưa :333.

TỔNG HỢP LẠI CÁC BƯỚC:

- B1: Tạo cây và liên tục hiệu chỉnh các lá lên các vị trí đỉnh phù hợp với nó.
- B2: Sau khi hiệu chỉnh đến chiếc lá cuối cùng thì bắt đầu hoán đổi vị trí giữa đỉnh cây và chiếc lá cuối cùng (chiếc lá nằm bên phải phía dưới cùng của cây).
- B3: Lấy đi chiếc lá cuối cùng trong cây (lấy đi phần tử cuối cùng trong cây). Thực chất bước này được thực hiện sau khi các bạn hoán đổi phần tử ở gốc cây và chiếc lá cuối cùng trong cây và giảm cái đoạn cần xét lại trong mảng.
- B4: Quay lại B1 và thực hiện theo trình tự. Nếu mà cây chỉ còn 1 phần tử thì kết thúc thuật toán

5. Sắp xếp nhanh (Quick sort)

Hehe, nãy giờ tìm hiểu các bạn cũng có thể thấy, các thuật toán đều được đặt những cái tên nghe phát biết ngay đúng không?

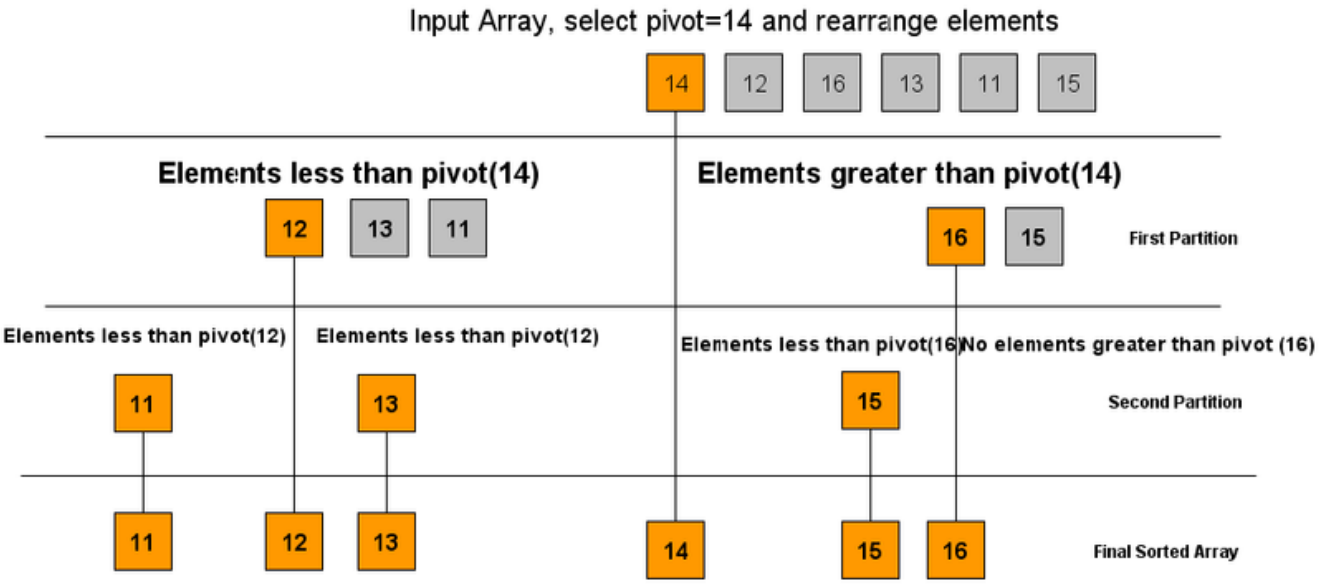
Tương tự, với quick sort, quick có nghĩa là nhanh, hứa hẹn mang đến cho các bạn một thuật toán có tốc độ sắp xếp cao hơn hẳn so với các thuật toán như Insertion sort, Bubble sort hay Selection sort

Quick sort có cùng cách thức hoạt động với Merge sort, đó là **"chia để trị"**

Tuy nhiên, đối với quick sort, có một bước siêu quan trọng, đó là phân đoạn thuật toán, bằng cách chọn một phần tử làm pivot. Có một số cách chọn pivot như sau:

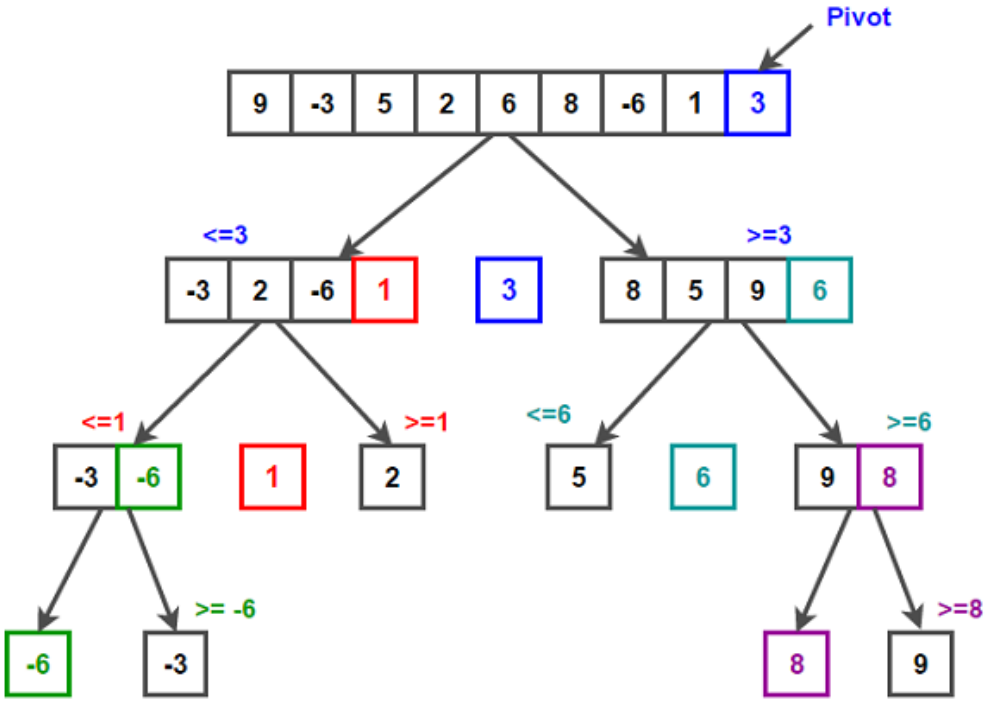
- Chọn phần tử đầu tiên của mảng
- Chọn phần tử cuối cùng của mảng
- Chọn một phần tử nằm giữa mảng
- Chọn một phần tử bất kì (**không khuyến khích**, trừ trường hợp bạn muốn tự tăng độ khó cho game, vì cách này rất dễ dẫn đến khả năng rơi vào các trường hợp đặc biệt, gây ra lặp vô hạn)

- Đối với pivot được chọn là phần tử đầu tiên của mảng:

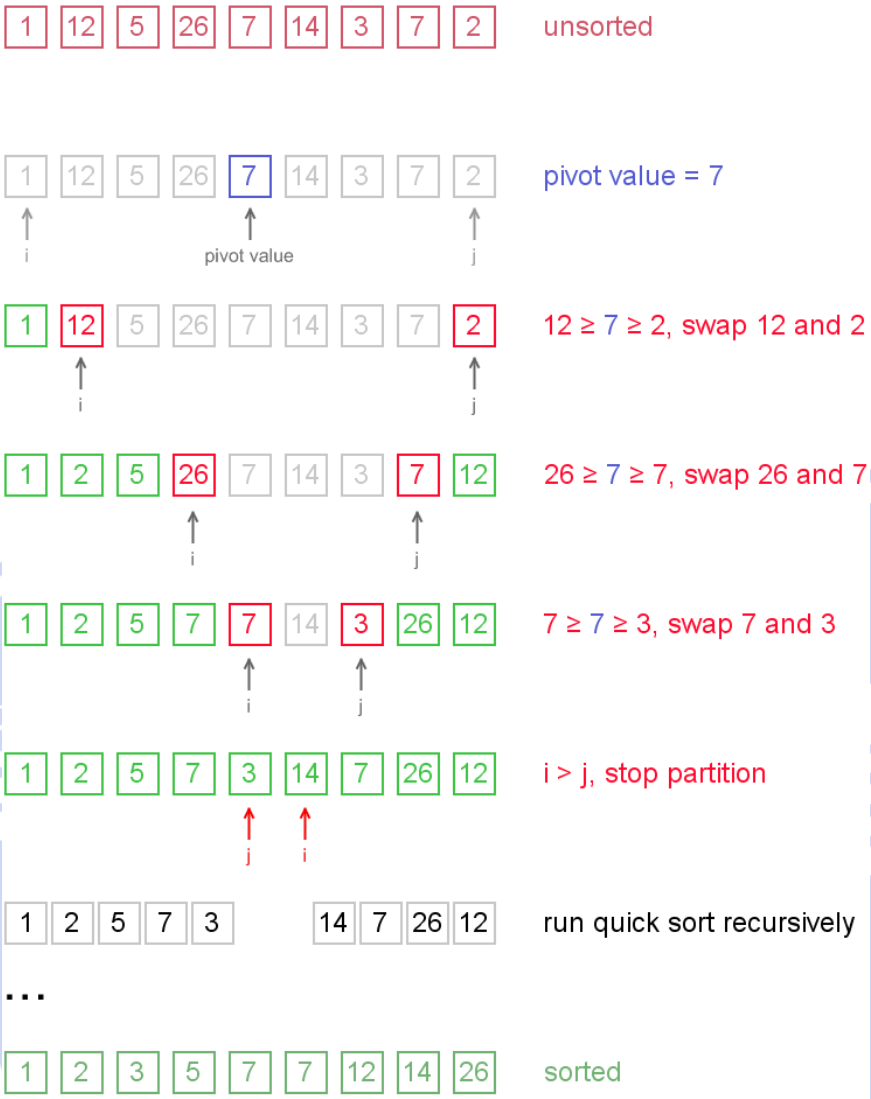


- Bước 1: Chọn phần tử đầu tiên làm chốt (14)
- Bước 2: Đưa những phần tử nhỏ hơn 14 vào mảng bên trái (12, 13, 11) và những phần tử lớn hơn 14 vào mảng bên phải (16, 15)
- Bước 3: thực hiện lại 2 bước trên với mảng bên trái và mảng bên phải

- Đối với pivot được chọn là phần tử cuối cùng của mảng:



- Đối với pivot được chọn là phần tử ở giữa mảng:



Do **hiệu quả của thuật toán** phụ thuộc rất nhiều vào việc **phần tử nào được chọn là phần tử chốt** nên độ phức tạp thuật toán cũng phụ thuộc vào đó mà khác nhau.

- Phân đoạn không cân bằng: không có phần nào cả, một bài toán con có kích thước n-1 và bài toán kia có kích thước là 0. Đó là trường hợp xấu nhất xảy ra khi dãy đã cho là dãy đã được sắp xếp và phần tử chốt được chọn là phần tử đầu của dãy => độ phức tạp thuật toán sẽ là O(n^2)
- Phân đoạn hoàn hảo: phân đoạn luôn thực hiện dưới dạng phân thì đôi, mỗi bài toán con có kích thước là n/2 => độ phức tạp thuật toán là O(nlogn)
- Phân đoạn cân bằng: một bài toán con có kích thước n-k và bài toán kia có kích thước là k => độ phức tạp thuật toán là O(n)

Để dễ hiểu hơn về cách thức mà thuật toán hoạt động, mời các bạn xem mô tả thuật toán (bao dễ hiểu ^^ coi một lần không hiểu? không thành vấn đề, hãy coi thêm lần nữa ^^):

https://www.youtube.com/watch?v=ywWBy6J5gz8

Tham khảo thêm: https://vi.wikipedia.org/wiki/S%E1%BA%AFp_x%E1%BA%BFp_nhanh

https://www.geeksforgeeks.org/quick-sort/

Tới đây là đã kết thúc phần 1 với những thuật toán sắp xếp thông dụng rồi. Các bạn hãy đón xem phần 2 với những những thuật toán khác và bài tập "hay ho" giúp các bạn rèn luyện tư duy giải thuật nha!

Phần 2 sẽ được lên sóng lúc **20h00 tối Chủ Nhật (23/05/2021)** trên kênh HTV3, à nhầm, trên fanpage của BHT, hẹn gặp các bạn nhaaaaa!

Phạm Bùi Nhật Huy – KHCL2020

Nguyễn Thị Như Vân – KHCL2020