Bài 2: Một số phương pháp sắp xếp

I. Thuật toán sắp xếp nhanh - Quick Sort

Ý tưởng:

Có dãy số: a1, a2, ..., an

Giải thuật QuickSort làm việc như sau:

Chọn **x** là một phần tử làm biên: thường chọn là phần tử ở giữa dãy số.

Phân hoạc dãy thành 3 dãy con

1. ak <= x , với k = 1..i
2. ak = x , với k = i..j
3. ak > =x , với k = j..N

Ak<= x	Ak=x	Ak>=x

Nếu số phần tử trong dãy con 1, 3 lớn hơn 1 thì ta tiếp tục phân hoạch dãy 1, 3 theo phương pháp trên. Ngược lại thì: dừng.

Giải thuật phân hoạch dãy am, am+1, ., an thành 2 dãy con:

Bước 1 : Chọn tùy ý một phần tử a[k] trong dãy là giá trị biên, m<= k <=n:

x = a[k]; i = m; j = n;

Bước 2 : Phát hiện và hiệu chỉnh cặp phần tử a[i], a[j] nằm sai vị trí:

Bước 2a : Trong khi (a[i]<x) i++;

Bước 2b : Trong khi (a[j]>x) j--;

Bước 2c : Nếu i<= j

// a[i]>= x; a[j]<=x mà a[j] đứng sau a[i]

Hoán vị (a[i],a[j]);

i++;

j--;

Bước 3 :

Nếu i <= j: Lặp lại Bước 2.//chưa xét hết mảng

Ngược lại: Dừng

Có thể phát biểu giải thuật sắp xếp QuickSort một cách đệ qui như sau :

Bước 1 : Phân hoạch dãy a_m … a_n thành các dãy con :

- Dãy con 1 : am.. aj <= x
- Dãy con 2 : aj+1.. ai-1 = x
- Dãy con 1 : ai.. a_n >= x

Bước 2 :

Nếu (m < j) // dãy con 1 có nhiều hơn 1 phần tử

Phân hoạch dãy a_m.. a_j

Nếu (i < n) // dãy con 3 có nhiều hơn 1 phần tử

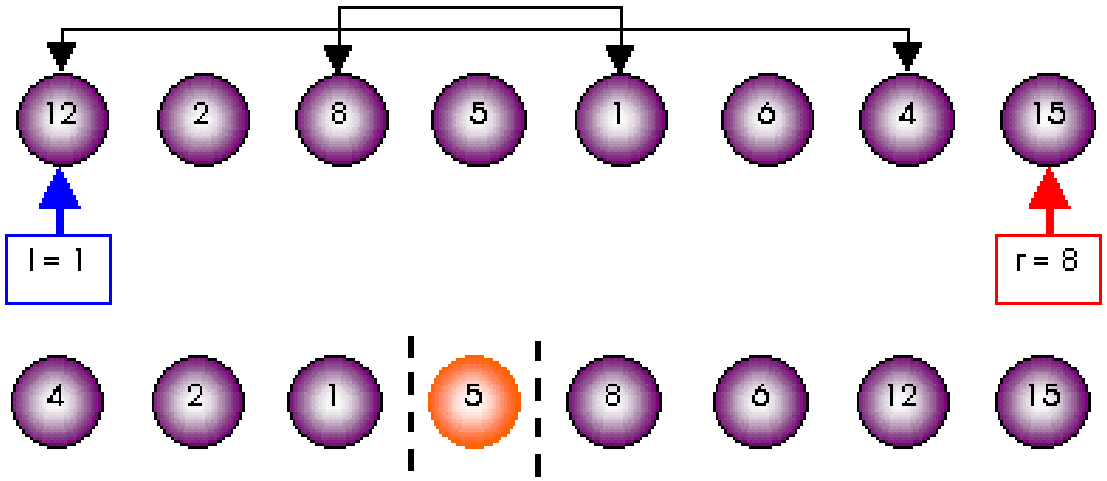
Phân hoạch dãy a_i.. a_r

Ví dụ:

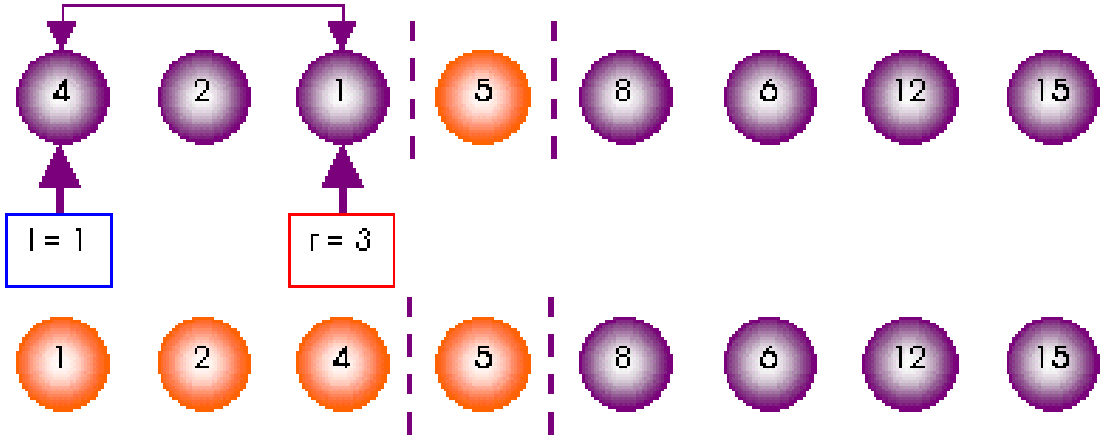
Cho dãy số a:

12 2 8 5 1 6 4 15

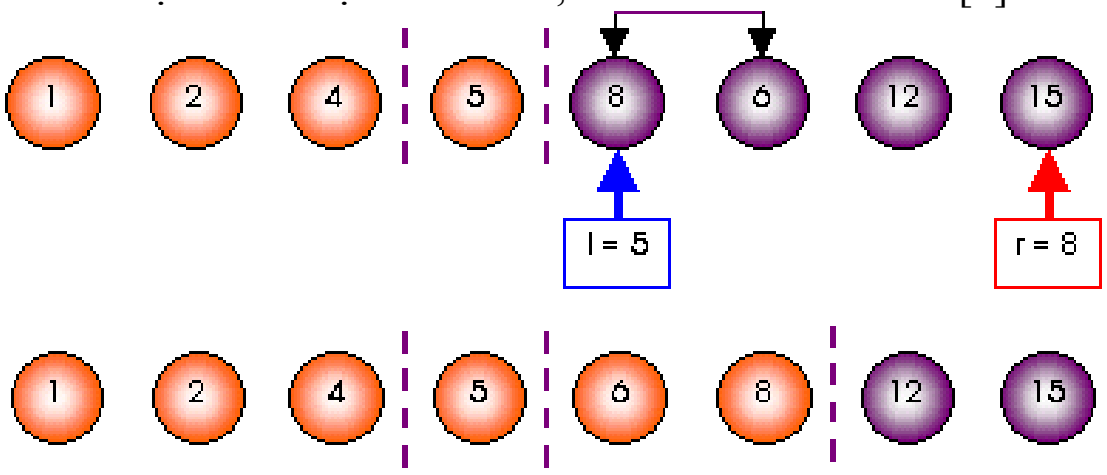
Phân hoạch đoạn ⌙ =1, r = 8: x = A[4] =5



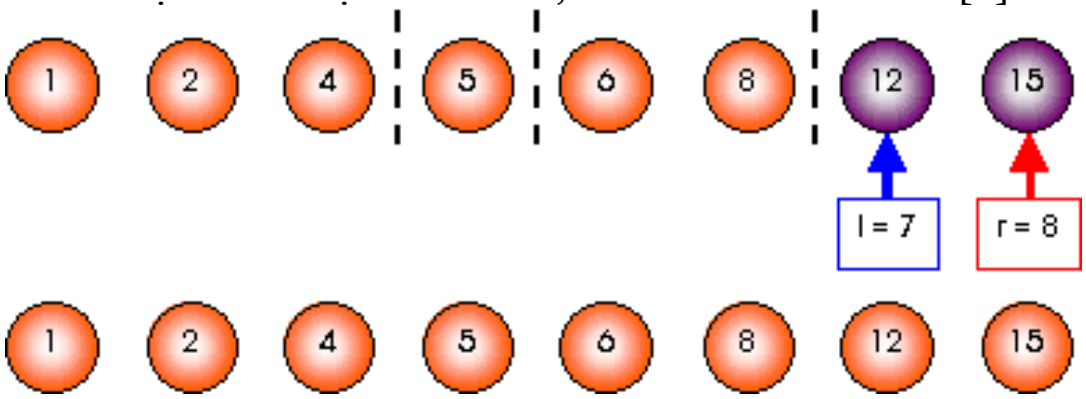
Phân hoạch đoạn l =1, r = 3: x = A[2] = 2



Phân hoạch đoạn l = 5, r = 8: x = A[6] = 6



Phân hoạch đoạn l = 7, r = 8: x = A[7] = 6



Dừng.

Cài đặt

Đánh giá giải thuật

Hiệu qủa thực hiện của giải thuật QuickSort phụ thuộc vào việc chọn giá trị mốc.

Trường hợp tốt nhất xảy ra nếu mỗi lần phân hoạch đều chọn được phần tử median (phần tử lớn hơn (hay bằng) nửa số phần tử, và nhỏ hơn (hay bằng) nửa số phần tử còn lại) làm mốc, khi đó dãy được phân chia thành 2 phần bằng nhau và cần $\log_2(n)$ bước phân hoạch thì sắp xếp xong.

Nhưng nếu mỗi bước phân hoạch phần tử được chọn có giá trị cực đại (hay cực tiểu) là mốc, dãy sẽ bị phân chia thành 2 phần không đều: một phần chỉ có 1 phần tử, phần còn lại gồm (n-1) phần tử, do vậy cần thực hiện n bước phân hoạch mới sắp xếp xong. Ta có bảng tổng kết

Trường hợp	Độ phức tạp
Tốt nhất	n*log(n)
Xấu nhất	n^2

II. Radix sort

Ý tưởng:

Khác với các thuật toán trước, Radix sort là một thuật toán tiếp cận theo một hướng hoàn toàn khác. Nếu như trong các thuật toán khác, cơ sở để sắp xếp luôn là việc so sánh giá trị của 2 phần tử thì Radix sort lại dựa trên nguyên tắc phân loại thư của bưu điện.

Ta biết rằng, để đưa một khối lượng thư lớn đến tay người nhận ở nhiều địa phương khác nhau, bưu điện thường tổ chức một hệ thống phân loại thư phân cấp:

Trước tiên, các thư đến cùng một tỉnh, thành phố sẽ được sắp chung vào một lô để gửi đến tỉnh thành tương ứng.

Bưu điện các tỉnh thành này lại thực hiện công việc tương tự. Các thư đến cùng một quận, huyện sẽ được xếp vào chung một lô và gửi đến quận, huyện tương ứng. Cứ như vậy, các bức thư sẽ được trao đến tay người nhận một cách có hệ thông mà công việc sắp xếp thư không quá nặng nhọc.

Mô phỏng lại qui trình trên, để sắp xếp dãy a1, a2, ..., an, giải thuật Radix Sort thực hiện như sau:

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.

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, ..

Các bước thực hiện thuật toán như sau:

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

Đặt 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

Ví dụ

Cho dãy số a:

701 1725 999 9170 3252 4518 7009 1424 428 1239 8425 7013

Phân lô theo hàng đơn vị:

12	0701										
11	1725										
10	0999										
9	9170										
8	3252										
7	4518										
6	7009										
5	1424										
4	0428										
3	1239										0999
2	8425						1725			4518	7009
1	7013	9170	0701	3252	7013	1424	8425			0428	1239
CS	A	0	1	2	3	4	5	6	7	8	9

Các lô B dùng để phân loại

Phân lô theo hàng chục:

12	0999										
11	7009										
10	1239										
9	4518										
8	0428										
7	1725										
6	8425										
5	1424										
4	7013			0428							
3	3252			1725							
2	0701	7009	4518	8425							
1	9170	0701	7013	1424	1239		3252		9170		0999
CS	A	0	1	2	3	4	5	6	7	8	9

Phân lô theo hàng trăm:

12	0999										
11	9170										
10	3252										
9	1239										
8	0428										
7	1725										
6	8425										
5	1424										

4	4518										
3	7013					0428					
2	7009	7013		3252		8425			1725		
1	0701	7009	9170	1239		1424	4518		0701		0999
CS	A	0	1	2	3	4	5	6	7	8	9

Phân lô theo hàng ngàn:

12	0999										
11	1725										
10	0701										
9	4518										
8	0428										
7	8425										
6	1424										
5	3252										
4	1239										
3	9170	0999	1725								
2	7013	0701	1424						7013		
1	7009	0428	1239		3252	4518			7009	8425	9170
CS	A	0	1	2	3	4	5	6	7	8	9

Lấy các phần tử từ các lô B0, B1, ., B9 nối lại thành a:

12	9170										
11	8425										
10	7013										
9	7009										
8	4518										

7	3252										
6	1725										
5	1424										
4	1239										
3	0999										
2	0701										
1	0428										
CS	A	0	1	2	3	4	5	6	7	8	9

Đánh giá giải thuật

Với một dãy n số, mỗi số có tối đa m chữ số, thuật toán thực hiện m lần các thao tác phân lô và ghép lô. Trong thao tác phân lô, mỗi phần tử chỉ được xét đúng một lần, khi ghép cũng vậy. Như vậy, chi phí cho việc thực hiện thuật toán hiển nhiên là O(2mn) = O(n).

NHẬN XÉT

Thuật toán không có trường hợp xấu nhất và tốt nhất. Mọi dãy số đều được sắp với chi phí như nhau nếu chúng có cùng số phần tử và các khóa có cùng chiều dài.

Thuật toán cài đặt thuận tiện với các mảng có khóa sắp xếp là chuỗi (ký tự hay số) hơn là khóa số như trong ví dụ do tránh được chi phí lấy các chữ số của từng số.

Tuy nhiên, số lượng lô nhiều (10 khi dùng số thập phân, 26 khi dùng chuỗi ký tự tiếng anh, ...) nhưng tổng kích thước của tất cả các lô chỉ bằng dãy ban đầu nên ta không thể dùng mảng để biểu diễn B (B0->B9). Như vậy, phải dùng cấu trúc dữ liệu động để biểu diễn B => Radix sort rất thích hợp cho sắp xếp trên danh sách liên kết.

Khi sắp các dãy không nhiều phần tử, thuật toán Radix sort sẽ mất ưu thế so với các thuật toán khác.

III. Sắp xếp cây - Heap sort

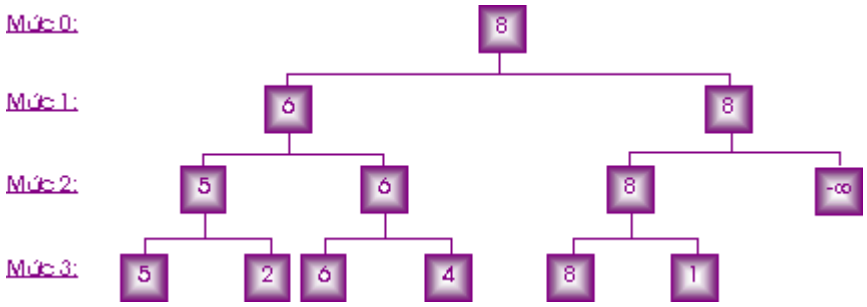
1.Ý tưởng:

Nhận xét: Khi tìm phần tử nhỏ nhất ở bước i, phương pháp sắp xếp chọn trực tiếp không tận dụng được các thông tin đã có được do các phép so sánh ở bước i-1.

Vì lý do trên người ta tìm cách xây dựng một thuật toán sắp xếp có thể khắc phục nhược điểm này.

Mấu chốt để giải quyết vấn đề vừa nêu là phải tìm ra được một cấu trúc dữ liệu cho phép tích lũy các thông tin về sự so sánh giá trị các phần tử trong qua trình sắp xếp.

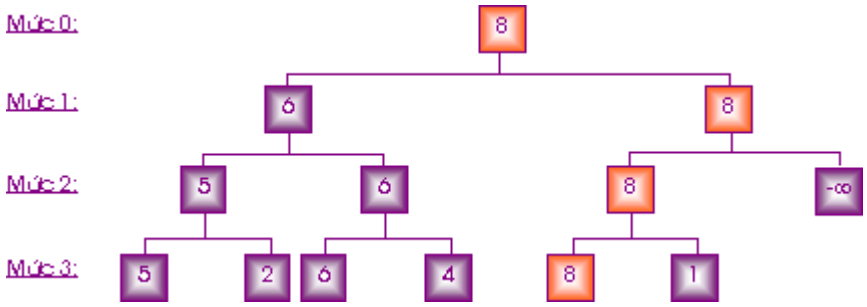
Giả sử dữ liệu cần sắp xếp là dãy số : 5 2 6 4 8 1 được bố trí theo quan hệ so sánh và tạo thành sơ đồ dạng cây như sau :



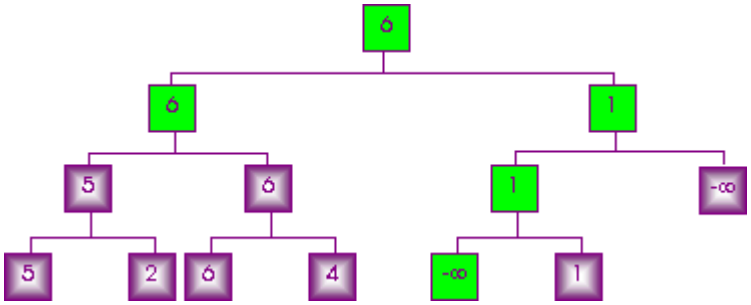
Trong đó một phần tử ở mức i chính là phần tử lớn trong cặp phần tử ở mức i+1, do đó phần tử ở mức 0 (nút gốc của cây) luôn là phần tử lớn nhất của dãy.

Nếu loại bỏ phần tử gốc ra khỏi cây (nghĩa là đưa phần tử lớn nhất về đúng vị trí), thì việc cập nhật cây chỉ xảy ra trên những nhánh liên quan đến phần tử mới loại bỏ, còn các nhánh khác được bảo toàn, nghĩa là bước kế tiếp có thể sử dụng lại các kết quả so sánh ở bước hiện tại.

Trong ví dụ trên ta có :



Loại bỏ 8 ra khỏi cây và thế vào các chỗ trống giá trị -∞ để tiện việc cập nhật lại cây :



Tiến hành nhiều lần việc loại bỏ phần tử gốc của cây cho đến khi tất cả các phần tử của cây đều là -∞, khi đó xếp các phần tử theo thứ tự loại bỏ trên cây sẽ có dãy đã sắp xếp. Trên đây là ý tưởng của giải thuật sắp xếp cây.

2. Cấu trúc dữ liệu Heap

Tuy nhiên, để cài đặt thuật toán này một cách hiệu quả, cần phải tổ chức một cấu trúc lưu trữ dữ liệu có khả năng thể hiện được quan hệ của các phần tử trong cây với n ô nhớ thay vì 2n-1 như trong ví dụ.

Khái niệm heap và phương pháp sắp xếp Heapsort do J.Williams đề xuất đã giải quyết được các khó khăn trên.

Định nghĩa Heap:

Giả sử xét trường hợp sắp xếp tăng dần, khi đó Heap được định nghĩa là một dãy các phần tử a_p, a_2 ,... , a_q thoả các quan hệ sau với mọi i thuộc [p, q]:

1/.	$a_i >= a_{2i}$
2/.	$a_i >= a_{2i+1}$ {(ai , a2i), (ai ,a2i+1) là các cặp phần tử liên đới }

Heap có các tính chất sau :

Tính chất 1 : Nếu ap , a2 ,... , aq là một heap thì khi cắt bỏ một số phần tử ở hai đầu của heap, dãy con còn lại vẫn là một heap.

Tính chất 2 : Nếu ap , a2 ,... , aq là một heap thì phần tử a1 (đầu heap) luôn là phần tử lớn nhất trong heap.

Tính chất 3 : Mọi dãy ap , a2 ,... , aq, dãy con aj, aj+1,…, ar tạo thành một heap với j=(q div 2 +1).

Giải thuật Heapsort :

Giải thuật Heapsort trải qua 2 giai đoạn :

Giai đoạn 1 :Hiệu chỉnh dãy số ban đầu thành heap;

Giai đoạn 2: Sắp xếp dãy số dựa trên heap:

Bước 1: Đưa phần tử lớn nhất về vị trí đúng ở cuối dãy: r = n; Hoánvị (a1 , ar);

Bước 2: Loại bỏ phần tử lớn nhất ra khỏi heap: r = r-1; Hiệu chỉnh phần còn lại của dãy từ a1 , a2 ... ar thành một heap.

Bước 3: Nếu r>1 (heap còn phần tử): Lặp lại Bước 2 Ngược lại : Dừng

Dựa trên tính chất 3, ta có thể thực hiện giai đoạn 1 bằng cách bắt đầu từ heap mặc nhiên an/2+1 , an/2+2 ... an, sau đó thêm

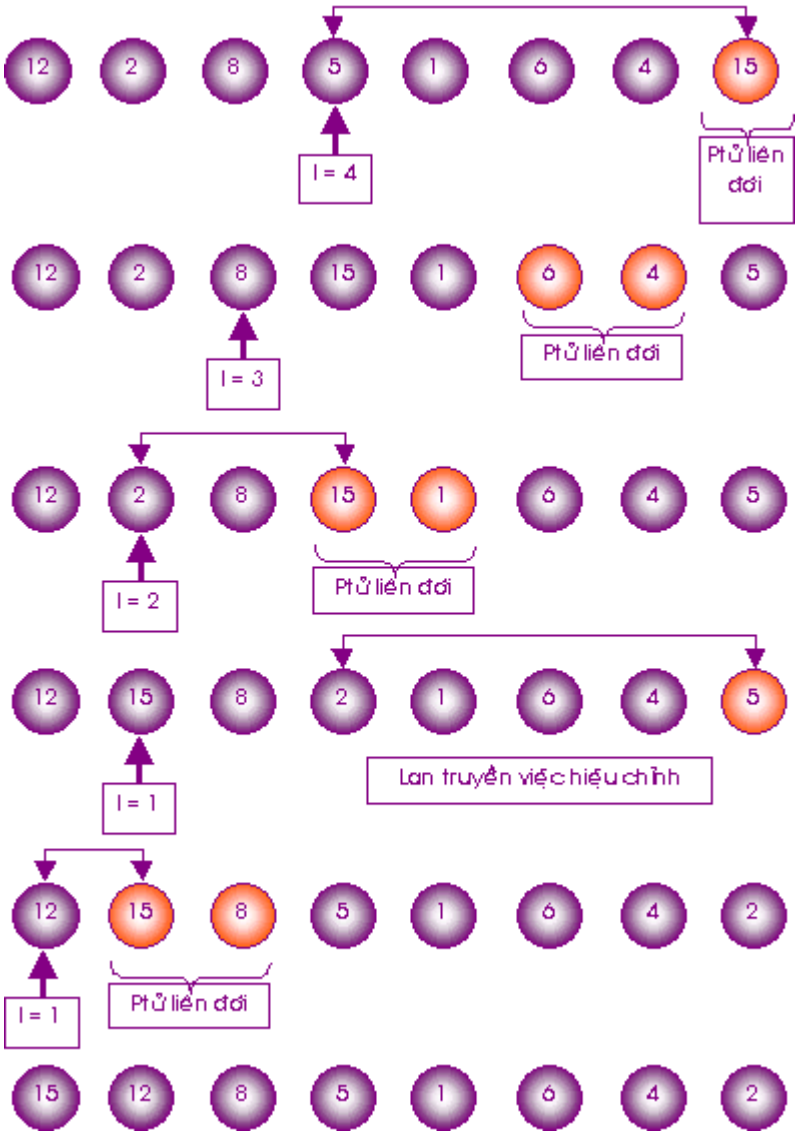
dần các phần tử an/2, an/2-1, ., a1 ta sẽ nhân được heap theo mong muốn.

Ví dụ

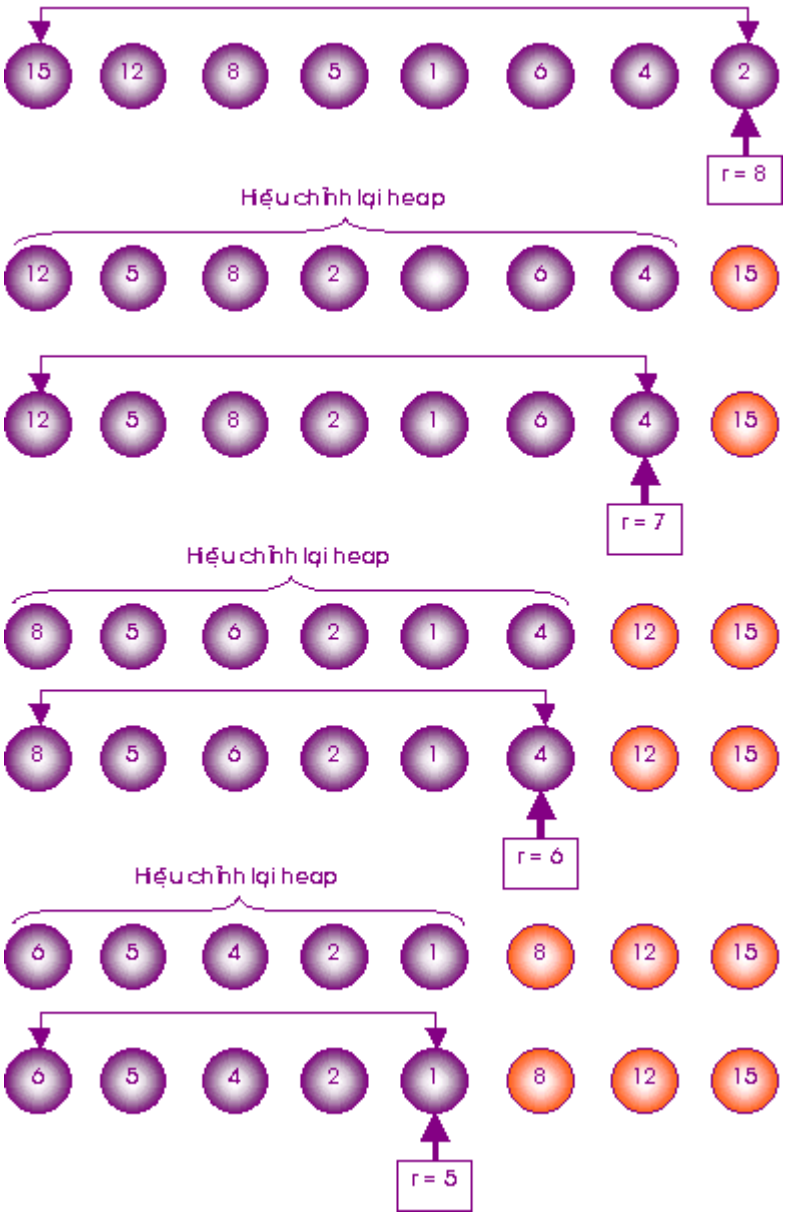
Cho dãy số a:

12 2 8 5 1 6 4 15

Giai đoạn 1: hiệu chỉnh dãy ban đầu thành heap



Giai đoạn 2: Sắp xếp dãy số dựa trên heap :



thực hiện tương tự cho r=5,4,3,2 ta được:

Cài đặt

Đánh giá giải thuật

Trong giai đoạn sắp xếp ta cần thực hiện n-1 bước mỗi bước cần nhiều nhất là $\log_2^{(n-1)}$, $\log_2^{(n-2)}$, … 1 phép đổi chỗi.

Như vậy độ phức tạp thuật toán Heap sort O(nlog2n)