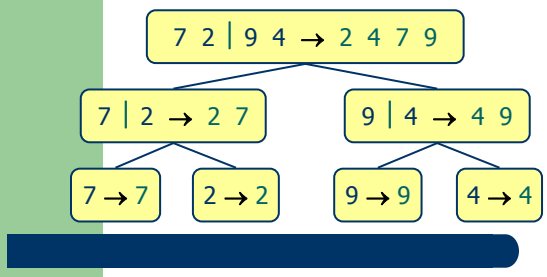
Cấu trúc dữ liệu và Giải thuật

Chương VI: Sắp xếp



Đỗ Bích Diệp - Khoa CNTT

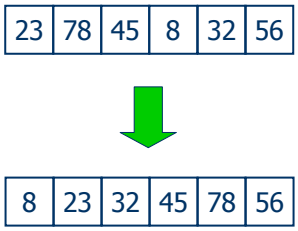
Chương VI: Sắp xếp

- Nội dung
  1. Bài toán sắp xếp
  2. Ba phương pháp sắp xếp cơ bản
     1. Lựa chọn, thêm dần và đổi chỗ
     2. Phân tích, đánh giá
  3. Sắp xếp kiểu hòa nhập
  4. Sắp xếp nhanh
  5. Sắp xếp kiểu vun đống
  6. Một số phương pháp sắp xếp đặc biệt

Đỗ Bích Diệp - Khoa CNTT

Bài toán Sắp xếp

- Sắp xếp lại một tập các phần tử dữ liệu theo chiều tăng dần hoặc giảm dần



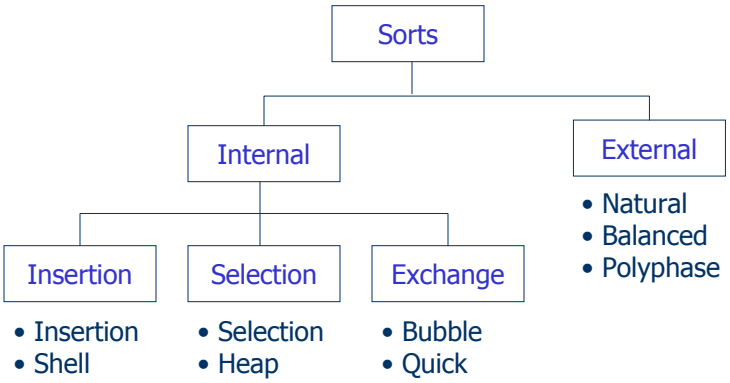
Đỗ Bích Diệp - Khoa CNTT

Bài toán Sắp xếp

- Khóa sắp xếp
  - Một bộ phận của bản ghi biểu diễn đối tượng được sắp
  - Khóa sẽ được sử dụng để xác định thứ tự sắp xếp bản ghi trong một tập các bản ghi
- Bảng khóa:
  - Sử dụng trong sắp xếp khi muốn hạn chế việc di chuyển các bản ghi dữ liệu
  - Một tập các bản ghi chỉ chứa hai trường
    - Khóa: chứa khóa sắp xếp
    - Link: Con trỏ ghi địa chỉ của bản ghi đối tượng dữ liệu tương ứng
  - Thứ tự các bản ghi trong bảng khóa cho phép xác định thứ tự của các bản ghi dữ liệu

Đỗ Bích Diệp - Khoa CNTT

Các loại thuật toán Sắp xếp



Đỗ Bích Diệp - Khoa CNTT

Bài toán Sắp xếp

- Các đặc trưng của thuật toán sắp xếp
  - Tính ổn định của thuật toán sắp xếp
    - Các phần tử có cùng khóa sẽ giữ nguyên thứ tự tương đối của chúng như trước khi sắp xếp

  - Tính tại chỗ
    - Thuật toán đòi hỏi không gian nhớ phụ là hằng số (không phụ thuộc vào số lượng phần tử trong dãy cần sắp)

Đỗ Bích Diệp - Khoa CNTT

Bài toán Sắp xếp

- Trong chương này, bài toán sắp xếp được đơn giản hóa dưới dạng như sau
  - Đầu vào: Một dãy các số nguyên $a_1, a_2, \ldots, a_n$
  - Đầu ra : Một hoán vị của dãy số đã cho trong đó các giá trị được sắp xếp theo chiều tăng dần

Đỗ Bích Diệp - Khoa CNTT

Ba phương pháp sắp xếp cơ bản

1. Sắp xếp kiểu lựa chọn (Selection Sort)
2. Sắp xếp kiểu thêm dần (Insertion Sort)
3. Sắp xếp kiểu đổi chỗ - Sắp xếp kiểu nổi bọt (Buble Sort)

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu lựa chọn – Selection Sort

- Ý tưởng:
  - Tại mỗi lượt, chọn phần tử nhỏ nhất trong số các phần tử chưa được sắp. Đưa phần tử được chọn vào vị trí đúng bằng phép đổi chỗ.
  - Sau lượt thứ i (i = 1..n-1) , dãy cần sắp coi như được chia thành 2 phần
    - Phần đã sắp: từ vị trí 1 đến i
    - Phần chưa sắp: từ vị trí i +1 đến n

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu lựa chọn

- Ví dụ: Sắp xếp dãy sau theo thứ tự tăng dần:
  - A = {12, 5, 3, 10, 18, 4, 9, 16}

	Lượt 1	Lượt 2	Lượt 3	Lượt 4	Lượt 5	Lượt 6	Lượt 7
12	3	3	3	3	3	3	3
5	5	4	4	4	4	4	4
3	12	12	5	5	5	5	5
10	10	10	10	9	9	9	9
18	18	18	18	18	10	10	10
4	4	5	12	12	12	12	12
9	9	9	9	10	18	18	16
16	16	16	16	16	16	16	18

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu lựa chọn

```
Procedure SELECTION-SORT(A,n)
1.  for i = 1 to n-1 do begin
2.  {Duyệt từ đỉnh}
      min = i;
3.  {Chọn phần tử nhỏ nhất}
          for j = i+1 to n do
            if A[j] < A[min] then
              min = j ;
4. {Đổi chổ phần tử i và phần tử nhỏ nhất}
     T = A[i];  A[i] = A[min]; A[min] = T;
   end;
End.
```

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu lựa chọn

- Thời gian thực hiện thuật toán
  - Trường hợp tốt nhất:
    - Dãy ban đầu đã được sắp xếp
    - 0 phép đổi chỗ, chỉ thực hiện n(n-1)/2 phép so sánh
  - Trường hợp xấu nhất
    - n-1 phép đổi chỗ, n(n-1)/2 phép so sánh
- Độ phức tạp thời gian trung bình $O(n^2)$

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu thêm dần – Insertion sort

- Ý tưởng:
  - Dãy cần sắp được chia thành 2 phần: một là phần đã sắp, còn lại là phần chưa sắp
  - Tại mỗi lượt, phần tử đầu tiên trong phần chưa sắp sẽ được "thêm" vào đúng vị trí của nó trong phần đã sắp.

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu thêm dần

- Ví dụ: Sắp xếp dãy sau theo thứ tự tăng dần:
  - A = {12, 5, 3, 10, 18, 4, 9, 16}

	Lượt 1	Lượt 2	Lượt 3	Lượt 4	Lượt 5	Lượt 6	Lượt 7
12	5	3	3	3	3	3	3
5	12	5	5	5	4	4	4
3	3	12	10	10	5	5	5
10	10	10	12	12	10	9	9
18	18	18	18	18	12	10	10
4	4	4	4	4	18	12	12
9	9	9	9	9	9	18	16
16	16	16	16	16	16	16	18

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu thêm dần

```
Procedure INSERTION-SORT(A,n)
1. for i := 2 to n do begin
2.  {Chọn phần tử đầu tiên của phần chưa được sắp xếp}
     val := A[i];
     j := i;
    {Tìm vị trí thích hợp để chèn phần tử A[i] trong phần đã sắp- chứa các phần tử từ
    vị trí 1 đến i-1}
     while ( j > 1) and (A[j-1] > val) do
     begin
          A[j] := A[j-1]; j := j -1;
     end;
4. {Chèn phần tử A[i] vào vị trí thích hợp}
    A[j] := val; end;
5. End
```

Sắp xếp kiểu thêm dần

- Sắp xếp thêm dần là tại chỗ và ổn định
- Thời gian thực hiện giải thuật
  - Trường hợp tốt nhất:
    - Dãy ban đầu đã được sắp xếp
    - 0 thực hiện phép đổi chỗ, n-1 phép so sánh
  - Trường hợp xấu nhất
    - n(n-1)/2 phép đổi chỗ và so sánh
- Độ phức tạp thời gian trung bình $O(n^2)$

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu nổi bọt

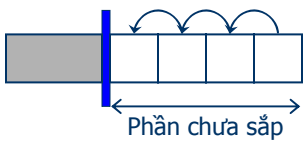
- Ví dụ
  - A = {12, 5, 3, 10, 18, 4, 9, 16}

	Lượt 1	Lượt 2	Lượt 3	Lượt 4	Lượt 5	Lượt 6	Lượt 7
12	3	3	3	3	3	3	3
5	12	4	4	4	4	4	4
3	5	12	5	5	5	5	5
10	4	5	12	9	9	9	9
18	10	9	9	12	10	10	10
4	18	10	10	10	12	12	12
9	9	18	16	16	16	16	16
16	16	16	18	18	18	18	18

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu nổi bọt

- Ý tưởng:
  - Dãy cần sắp được chia thành 2 phần: một là phần đã sắp, còn lại là phần chưa sắp
  - Thông qua phép đổi chỗ, tại mỗi lượt phần tử nhỏ nhất trong phần chưa được sắp sẽ được "đẩy dần" lên trước và cuối cùng nhập vào phần được sắp.



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu nổi bọt

```
Procedure BUBBLE-SORT(A,n)
1.  for i := 1 to n-1 do
2.  {Duyệt từ đáy}
        for j:= n down to i+1 do
3.  {Kiểm tra 2 phần tử kề cận nhau, nếu ngược thứ tự thì đổi  chỗ }
            if A[j] < A[j-1] then
              begin
                  X:= A[j];
                  A[j] := A[j-1];
                  A[j-1] := X;
               end
4.  return
```

Đỗ Bích Diệp - Khoa CNTT

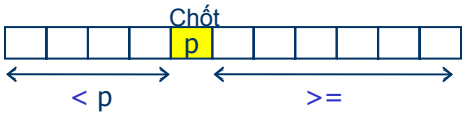
Sắp xếp kiểu nổi bọt

- Thời gian thực hiện giải thuật
  - Trường hợp tốt nhất:
    - Dãy ban đầu đã được sắp xếp
    - 0 thực hiện phép đổi chỗ, n(n-1)/2 phép so sánh
  - Trường hợp xấu nhất
    - n(n-1)/2 phép đổi chỗ và so sánh
- Độ phức tạp thời gian trung bình $O(n^2)$

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp Nhanh (Quick Sort)

- Được đưa ra bởi C. A. Hoare (1962).
- Là phương pháp sắp xếp dựa trên chiến lược chia để trị
  - **Trường hợp cơ sở**: Dãy chỉ có 1 phần tử, dãy đã được sắp
  - **Chia** – Pha phân đoạn
    - Chọn một phần tử trong dãy làm phần tử chốt p
    - Chia dãy đã cho thành 3 nhóm : <p ; p ; >=p



  - **Trị**:
    - Sắp xếp được tiếp tục một cách đệ qui với nhóm thứ 1 và nhóm thứ 3

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp nhanh

```
Procedure QUICK-SORT(A, left, right)
{A là mảng cần sắp, left là chỉ số của phần tử đầu , right là chỉ số của phần tử cuối}
1.  if left < right then begin
        p = PARTITION(A,left, right) ;
        QUICK-SORT(A, left, p-1);
        QUICK-SORT(A, p+1,right);
                end;
2.  return.
```

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp nhanh

- Pha phân đoạn – Partition
  - Hàm Partition thực hiện chia dãy đầu vào A[left..right] thành 2 đoạn
    - A[left, p-1] gồm các phần tử nhỏ hơn hoặc bằng A[p]
    - A[p+1, right] gồm các phần tử lớn hơn hoặc bằng A[p]
  - Gồm hai công đoạn chính
    - Lựa chọn chốt
    - Thực hiện Phân đoạn

Đỗ Bích Diệp - Khoa CNTT

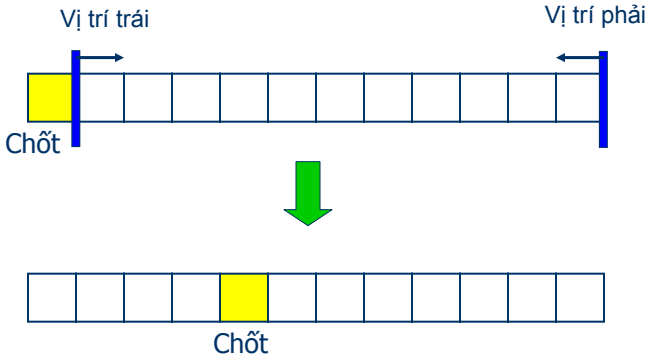
Sắp xếp nhanh

- Lựa chọn chốt
  - Chọn chốt là phần tử đứng đầu hoặc cuối danh sách
  - Chọn phần tử đứng giữa danh sách làm chốt
  - Chọn phần tử trung vị trong 3 phần tử đứng đầu, đứng giữa và đứng cuối danh sách
  - Chọn phần tử ngẫu nhiên

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp nhanh

- Phân đoạn



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp nhanh

Function PARTITION-LEFT(A, left, right)

{A là mảng cần sắp, left là chỉ số của phần tử đầu , right là chỉ số của phần tử cuối. Phần tử chốt là phần tử ở đầu danh sách}

1. i:=left + 1; j := right; pivot = left // i là khởi đầu của vị trí trái, j là khởi đầu của vị trí phải
2. { Tiến hành duyệt, so sánh, đổi chỗ để hình thành phân đoạn}

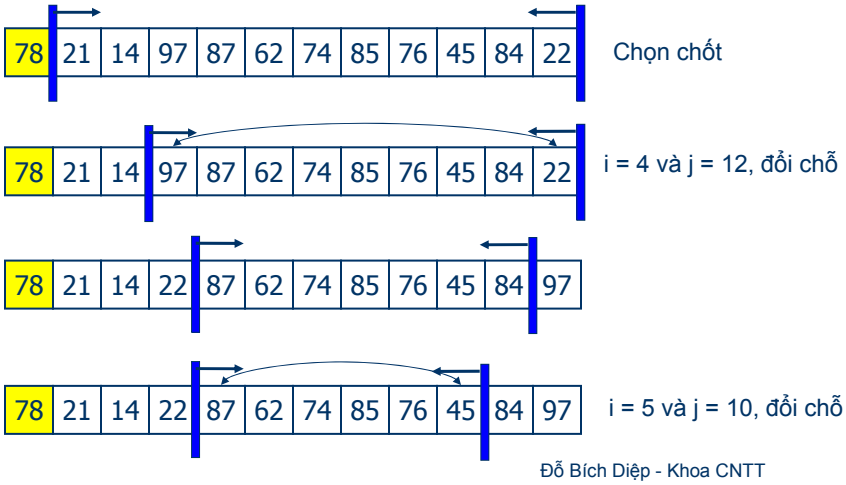
```
while ( i<=j) do begin
    while (A[i] < A[pivot]) do i := i+1;
    while (A[j] > A[pivot]) do j:= j-1;
     if i < j then begin A[i] <-> A[j]; i := i+1; j := j -1; end
end
```

3. {Đưa chốt về vị trí thực giữa 2 phân đoạn, lưu vị trí thực của phần tử chốt}

```
k:= j; A[pivot] <-> A[j];
```

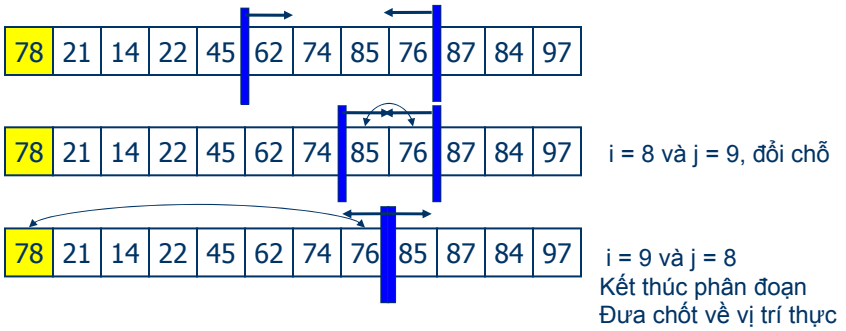
4. Return k

Sắp xếp nhanh



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp nhanh



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp nhanh

```
Function PARTITION-MID(A, left, right)
{A là mảng cần sắp, left là chỉ số của phần tử đầu , right là chỉ số của phần tử cuối.
Phần tử chốt là phần tử ở đầu danh sách}
1.  i:=left ; j := right; pivot = [(left + right ) /2 ]
{pivot là số nguyên >= (left+right)/2}
2.  repeat
        while (A[i] < A[pivot]) do i := i+1;
        while (A[j] > A[pivot]) do j:= j-1;
        if i <= j then begin A[i] <-> A[j]; i := i+1; j := j -1; end
    until i > j
4.  Return j
```

Đỗ Bích Diệp - Khoa CNTT

Đánh giá giải thuật Sắp xếp nhanh

- Sắp xếp nhanh là tại chỗ nhưng không ổn định
- Thời gian thực hiện giải thuật
  - Trường hợp tổng quát
    - T(0) = T(1) = c
    - Pha phân đoạn được thực hiện bằng việc duyệt danh sách ban đầu 1 lần → Thời gian thực hiện là O(n)
    - Trong giải thuật xuất hiện 2 lời gọi đệ qui: Giả sử sau khi phân đoạn, phần tử chốt ở vị trí p thì
      T(n) = T(p-1) + T(n-p) + O(n) + O(1)

Đỗ Bích Diệp - Khoa CNTT

Đánh giá giải thuật Sắp xếp nhanh

- Trường hợp xấu nhất:
  - Công thức đệ qui: T(n) = T(n-1) + O(n) + O(1)
  - Độ phức tạp của giải thuật sắp xếp nhanh là $O(n^2)$ khi A vốn đã được sắp và chốt được chọn là nút nhỏ nhất
- Trường hợp hoàn hảo:
  - Phân đoạn cân bằng T(n) = 2 T(n/2) + n
  - Độ phức tạp trung bình của giải thuật là $O(n\log_2 n)$

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập

- Tương tự như sắp xếp nhanh dựa vào cơ chế chia để trị để thực hiện sắp xếp.
- Bao gồm 3 bước
  - Chia: Phân chia dãy cần được sắp S gồm n phần tử thành 2 dãy con với số phần tử là n/2 S_1 và S_2
  - Tri: Lần lượt sắp xếp hai dãy con S_1 và S_2 bằng sắp xếp kiểu hòa nhập
  - Tổ hợp: Nhập 2 dãy con đã được sắp S_1 và S_2 thành một dãy duy nhất

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập

Algorithm MERGE-SORT(S, n)

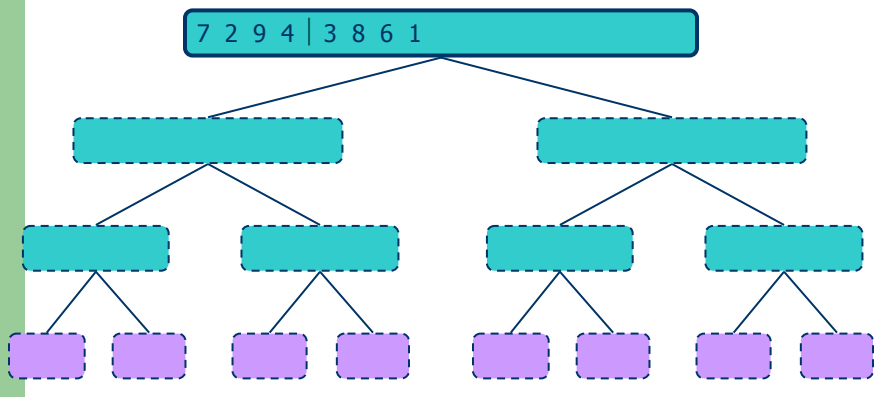
{S là dãy cần được sắp xếp, n là số phần tử trong dãy}

1. if (n< 2) then return S;
2. {Chia: Tạo dãy S1 chứa n div 2 phần tử đầu tiên của S, Tạo dãy S2 chứa các phần tử còn lại trong S sau khi đã lấy ra các phần tử trong S1}
   (S1, S2) = PARTITION(S, n div 2)
3. {Lặp}
   1. MERGE-SORT(S1, (n div 2));
   2. MERGE-SORT(S2, (n- (n div 2));
4. {Trị- Hòa nhập hai dãy được sắp }
   MERGE(S1,S2, S);
5. Return S;

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập – Ví dụ minh họa

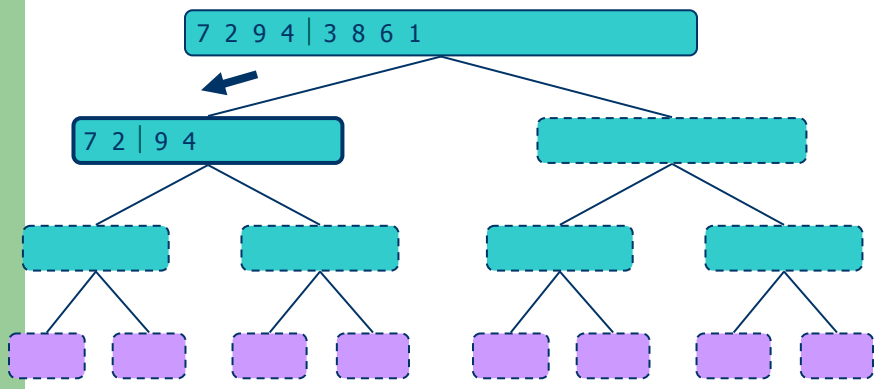
- Chia



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

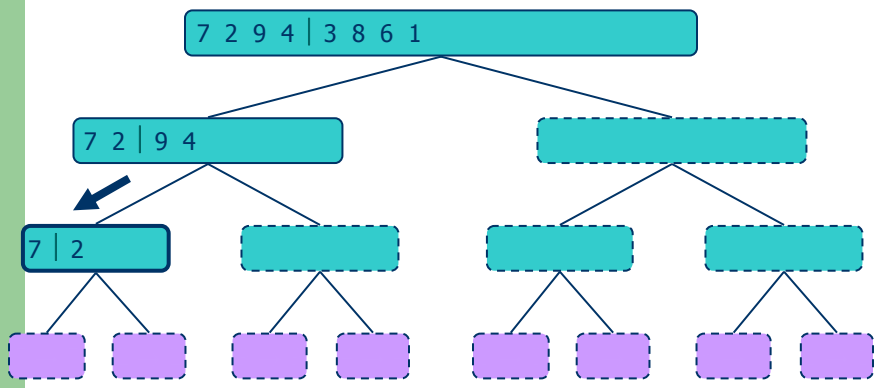
- Lời gọi đệ qui - Chia



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

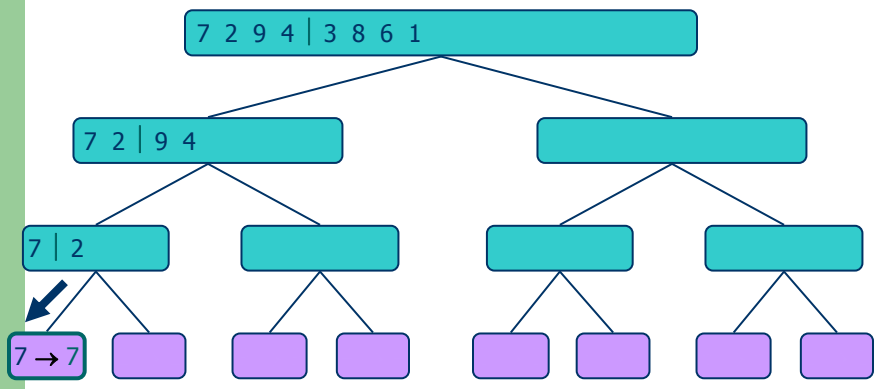
- Lời gọi đệ qui - Chia



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

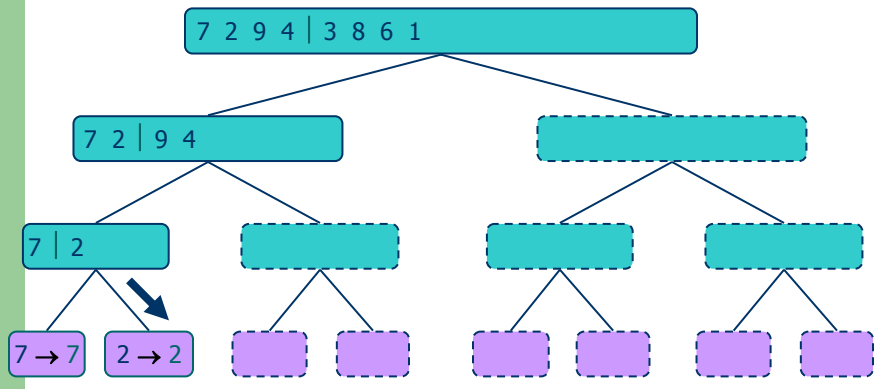
- Lời gọi đệ qui – Trường hợp cơ sở



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

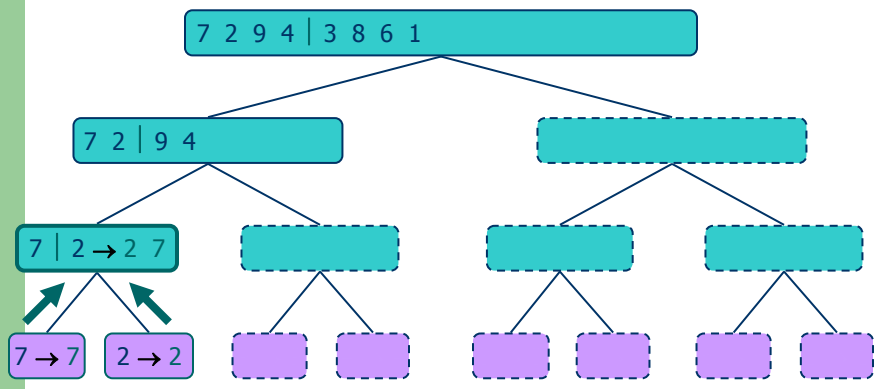
- Lời gọi đệ qui – Trường hợp cơ sở



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

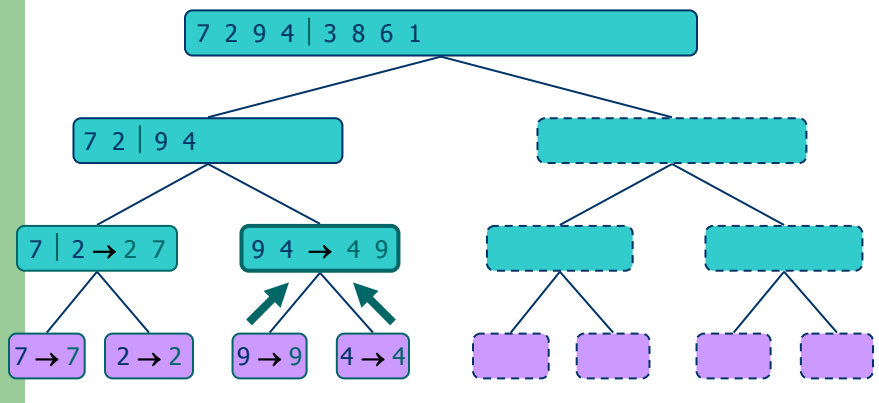
- Hòa nhập



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

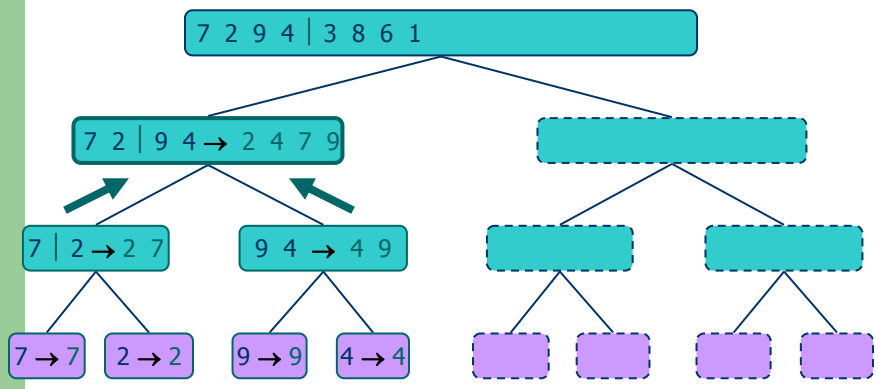
- Lời gọi đệ qui …. Trường hợp cơ sở , Hòa nhập



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

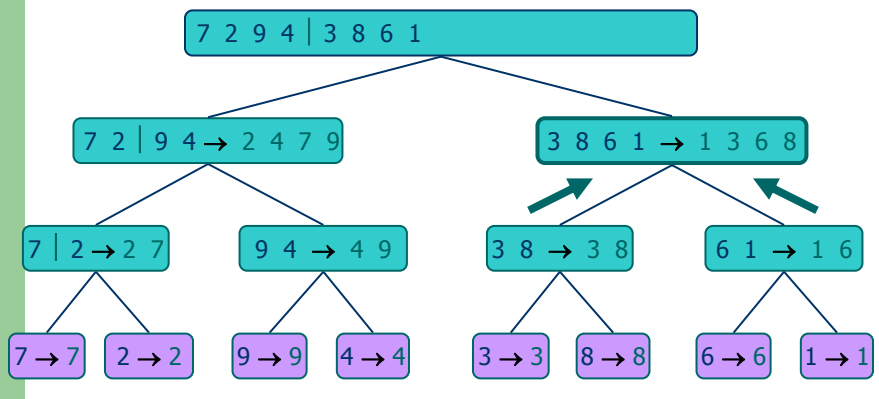
- Hòa nhập



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

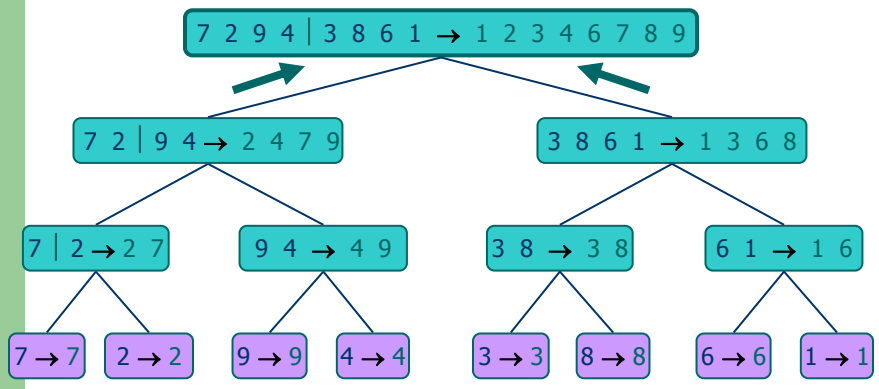
- Tương tự ….



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập - Ví dụ minh họa

- Hòa nhập lần cuối



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu hòa nhập

- Giải thuật: Hòa nhập hai dãy đã được sắp xếp

```
Procedure MERGE(A, B, C)
{A, B là hai dãy đã sắp với số phần tử lần lượt là sizea và sizeb, C là dãy hợp nhất
    của A và B}
1.  i:= 1; j:=1; k:=1 ; {khởi tạo các chỉ số trên 3 dãy A,B,C}
2.  { Tiến hành duyệt A và B, duyệt song song hai dãy cho đến khi một trong hai
    dãy kết thúc }
    while ( i<=sizea  and j <= sizeb) do
        if A[i] < B[j] then begin
                C[k] := A[i]; i := i+1; k:= k+1;
        end;
        else begin C[k] := B[j] ; j := j+1; k:= k+1 ;
        end;
    end;
```

Sắp xếp kiểu hòa nhập

```
3.  { Nếu dãy A hết }
    if i > sizea {dãy A đã hết} then for t:= 0 to sizeb – t do C[k+t] := B[j+t];
4.  else { dãy B hết} for t:= 0 to sizea – t do C[k+t] := A[i+t];
5.  return.
```

Đỗ Bích Diệp - Khoa CNTT

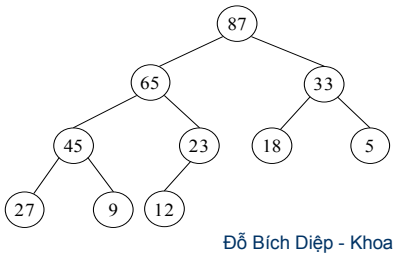
Sắp xếp kiểu hòa nhập

- Thời gian thực hiện giải thuật
  - $T(n) = 2\,T(n/2) + n$
- Độ phức tạp trong tình huống xấu nhất và trung bình là $O(n \log_2 n)$

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

- Cấu trúc Đống
  - Đống là một cây nhị phân có hai tính chất
    - Là cây nhị phân hoàn chỉnh
    - Có thứ tự : mỗi nút được gắn với một giá trị số tự nhiên, sao cho giá trị của nút cha bao giờ cũng lớn hơn giá trị của nút con (Max Heap)



Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

- Đống được lưu trữ trong máy tính dưới dạng một vector lưu trữ

87	65	33	45	23	18	5	27	9	12
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]	V[8]	V[9]	V[10]

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

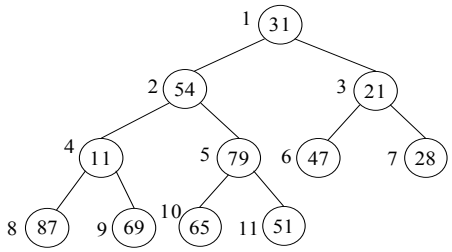
- Phép tạo đống
  - Dãy số cần sắp được coi là dãy các phần tử của một cây nhị phân hoàn chỉnh được lưu trữ kế tiếp
    - Dãy số A: {31, 54, 21, 11, 79, 47, 28, 87, 69, 65, 51}
    - Vector lưu trữ

31	54	21	11	79	47	28	87	69	65	51
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]	V[8]	V[9]	V[10]	V[11]

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

- Cây nhị phân hoàn chỉnh tương ứng



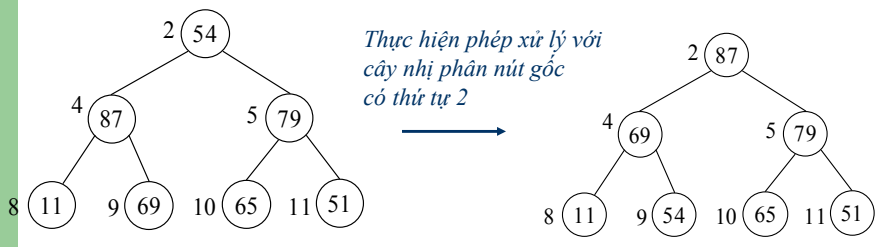
Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

- Hai thao tác cần thực hiện
  - Khôi phục tính chất đống của một nhánh cây có gốc là nút thứ i và hai con đã là đống
  - Xây dựng đống tương đương với một cây nhị phân hoàn chỉnh chưa phải là đống
    - Với lần lượt các cây con có gốc từ $\lfloor n/2 \rfloor$ xuống đến 1, khôi phục tính chất đống với các cây đó

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống



Khôi phục tính chất đống cho một cây con bất kỳ

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

Procedure BUILD-HEAP(i,n)

{Tạo đống trên cây có gốc là nút có thứ tự i trong n nút ban đầu }

1. VAL:= V[i]; {lưu giá trị của nút gốc của cây đang xét}

   j := 2*i; { j là số thứ tự của con trái của nút i}

2. while j <= n do begin

   if j <n and V[j] < V[j+1] then j:= j+1; {tìm chỉ số của nút con lớn hơn trong nút con bên phải và bên trái}

   if VAL >= V[j] then return; {khóa cha lớn hơn khóa con lớn nhất – đã có đống , không cần làm gì thêm}

   $V[\lfloor j/2 \rfloor] \leftrightarrow V[j]$; { đổi chỗ cha và con lớn nhất}

   j:= 2*j ; {đi xuống theo cây}

   end;

3. return

Sắp xếp kiểu vun đống

– Xây dựng đống với một cây gồm n nút

for i:=$\lfloor n/2 \rfloor$ down to 1 do call BUILD-HEAP(i,n);

Sắp xếp kiểu vun đống

- Sắp xếp kiểu vun đống: Chia làm 2 giai đoạn
  - Giai đoạn tạo đống ban đầu
  - Giai đoạn sắp xếp (Thực hiện n-1 lần với dãy gồm n số)
    - Đổi chỗ
    - Vun đống mới cho một dãy với ít hơn 1 phần tử so với đống trước

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

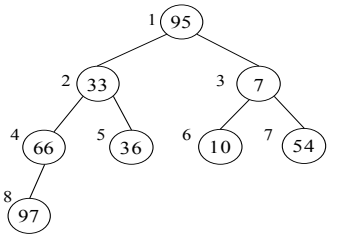
- Giải thuật sắp xếp kiểu vun đống

```
Procedure HEAP-SORT(V,n)
1.  {Tạo đống ban đầu}
    for i:=⌊n/2⌋ down to 1 do call BUILD-HEAP(i,n);
2.  {Sắp xếp}
    for i := n-1 down to 1 do begin
            V[1] ↔ V[i+1];
            call BUILD-HEAP(1,i)
    end;
3. return.
```

Đỗ Bích Diệp - Khoa CNTT

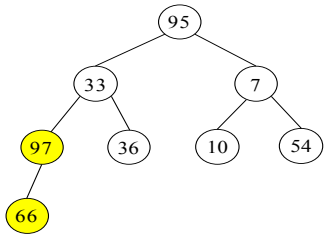
Sắp xếp kiểu vun đống

Giai đoạn tạo đống ban đầu



95	33	7	66	36	10	54	97

Cây và vector lưu trữ ban đầu

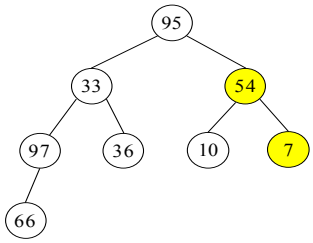


95	33	7	97	36	10	54	66

Sau khi thực hiện BUILD-HEAP(4,8)

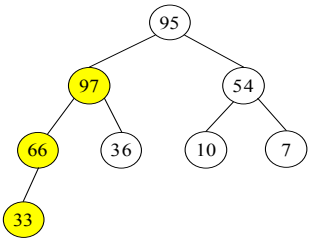
Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống



95	33	54	97	36	10	7	66

Sau khi thực hiện BUILD-HEAP(3,8)

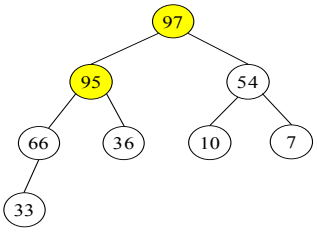


95	97	54	66	36	10	7	33

Sau khi thực hiện BUILD-HEAP(2,8)

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống



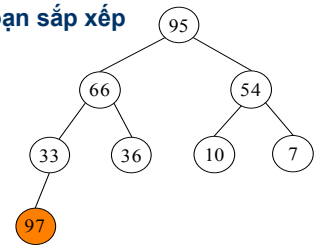
97	95	54	66	36	10	7	33

Sau khi thực hiện BUILD-HEAP(1,8). Hoàn thành việc tạo đống cho cây nhị phân hoàn chỉnh ban đầu

Đỗ Bích Diệp - Khoa CNTT

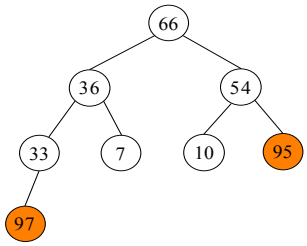
Sắp xếp kiểu vun đống

Giai đoạn sắp xếp



95	66	54	33	36	10	7	97

Sau khi đổi chỗ lần 1 giữa V[1] và V[8] và vun thành đống cho cây có 7 nút, số 97 đã vào đúng vị trí trong dãy

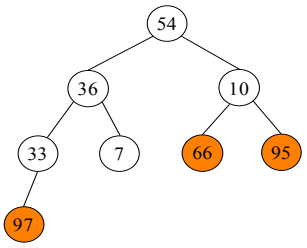


66	36	54	33	7	10	95	97

Sau khi đổi chỗ lần 2 giữa V[1] và V[7], vun thành đống cho cây có 6 nút, số 95 đã vào đúng vị trí trong dãy

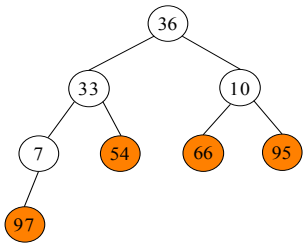
Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống



54	36	10	33	7	66	95	97

Sau khi đổi chỗ lần 3 giữa V[1] và V[6], vun thành đống cho cây có 5 nút, số 66 đã vào đúng vị trí trong dãy

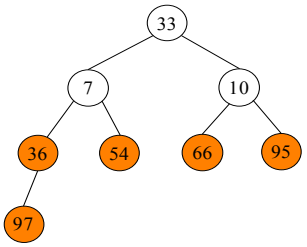


36	33	10	7	54	66	95	97

Sau khi đổi chỗ lần 4 giữa V[1] và V[5], vun thành đống cho cây có 4 nút, số 54 đã vào đúng vị trí trong dãy

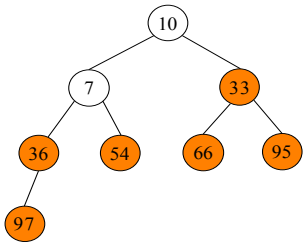
Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống



33	7	10	36	54	66	95	97

Sau khi đổi chỗ lần 5 giữa V[1] và V[4], vun thành đống cho cây có 3 nút, số 36 đã vào đúng vị trí trong dãy

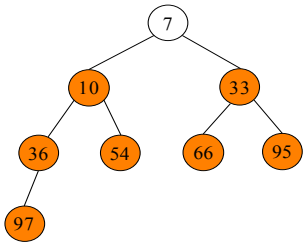


10	7	33	36	54	66	95	97

Sau khi đổi chỗ lần 6 giữa V[1] và V[3], vun thành đống cho cây có 2 nút, số 33 đã vào đúng vị trí trong dãy

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống



7	10	33	36	54	66	95	97

Đổi chỗ lần cuối cùng, bây giờ dãy số đã cho đã được sắp xếp theo thứ tự tăng dần

Đỗ Bích Diệp - Khoa CNTT

Sắp xếp kiểu vun đống

- Nhận xét, đánh giá giải thuật Sắp xếp kiểu vun đống
  - Thời gian thực hiện trung bình $T_{tb}(n) = O(n\log_2 n)$
  - Thời gian thực hiện trong trường hợp xấu nhất
    - Ở giai đoạn 1 có $\lfloor n/2 \rfloor$ lần gọi thủ tục BUILD-HEAP(i,n)
    - Ở giai đoạn 2 có n-1 lần gọi thực hiện thủ tục đó
    - THủ tục BUILD-HEAP được thực hiện trên một cây nhị phân hoàn chỉnh tối đa có n nút tức là có chiều cao h = $\log_2 n$, vậy thì số lượng phép so sánh cũng chỉ xấp xỉ $\log_2 n$. Vậy thời gian thực hiện BUILD-HEAP là $O(\log_2 n)$
    - Thời gian thực hiện HEAP-SORT trong trường hợp xấu nhất: $T_x(n) = 3n/2 * \log_2 n = O(n\log_2 n)$

Đỗ Bích Diệp - Khoa CNTT

Độ phức tạp của các phương pháp sắp xếp

Thuật giải	Average Case	Worst Case
Lựa chọn	$O(n^2)$	$O(n^2)$
Thêm dần	$O(n^2)$	$O(n^2)$
Đổi chỗ	$O(n^2)$	$O(n^2)$
Vun đống	O(nlogn)	O(nlogn)
Hòa nhập	O(nlogn)	O(nlogn)

Đỗ Bích Diệp - Khoa CNTT