BÀI 3

CÁC PHƯƠNG PHÁP TÌM KIẾM VÀ SẮP XẾP

3.2.5. SẮP XẾP NÂNG CAO

3.2.5. Sắp xếp nâng cao

- Sắp xếp phân đoạn (*Quick Sort*)
- Sắp xếp vun đống (*Heap Sort*)
- Sắp xếp trộn (*Merge Sort*)

3.2.5.1. Sắp xếp phân đoạn – quick sort

- **Ý tưởng**
  - **Phương pháp chia để trị**
    - Chia bài toán thành các bài toán con.
    - Giải quyết các bài toán con.
    - Tổng hợp kết quả.
  - **Kỹ thuật đệ quy**
    - Bài toán lớn được giải quyết nhờ việc giải quyết bài toán nhỏ cùng dạng nhưng có kích thước nhỏ hơn.

Sắp xếp phân đoạn (tt)

- **Ví dụ: Sắp xếp dãy số nguyên theo chiều tăng dần**

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]
53	-21	33	68	40	82	31	67	25

- Bài toán lớn được chia thành ba bài toán con.

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]
25	-21	33	31	40	82	68	67	53

- Trị ba bài toán con theo cách trên.
- Tổng hợp lời giải ta sẽ có dãy được sắp.

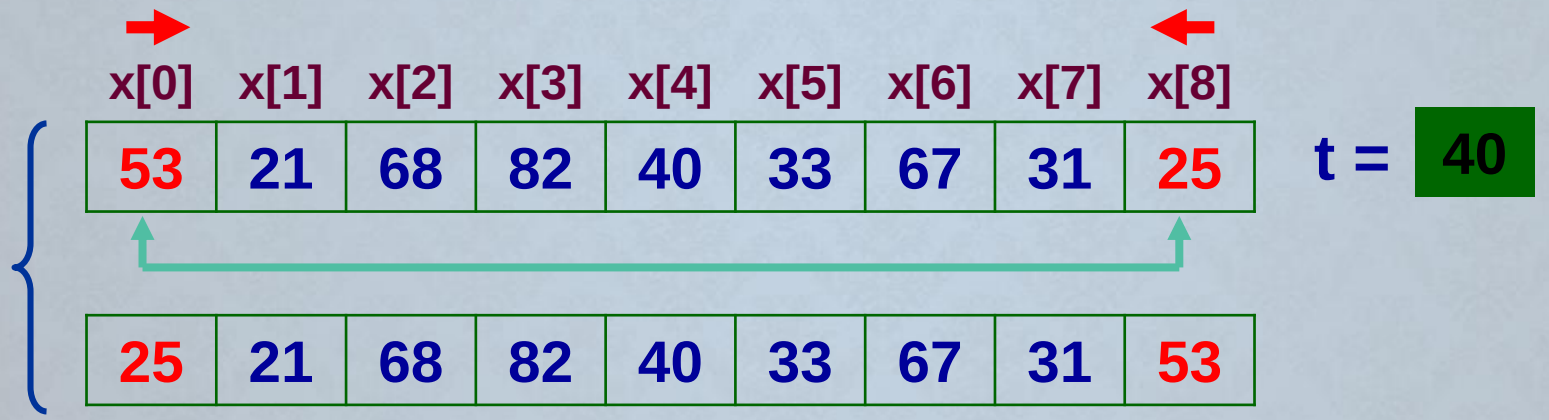
Sắp xếp phân đoạn (tt)

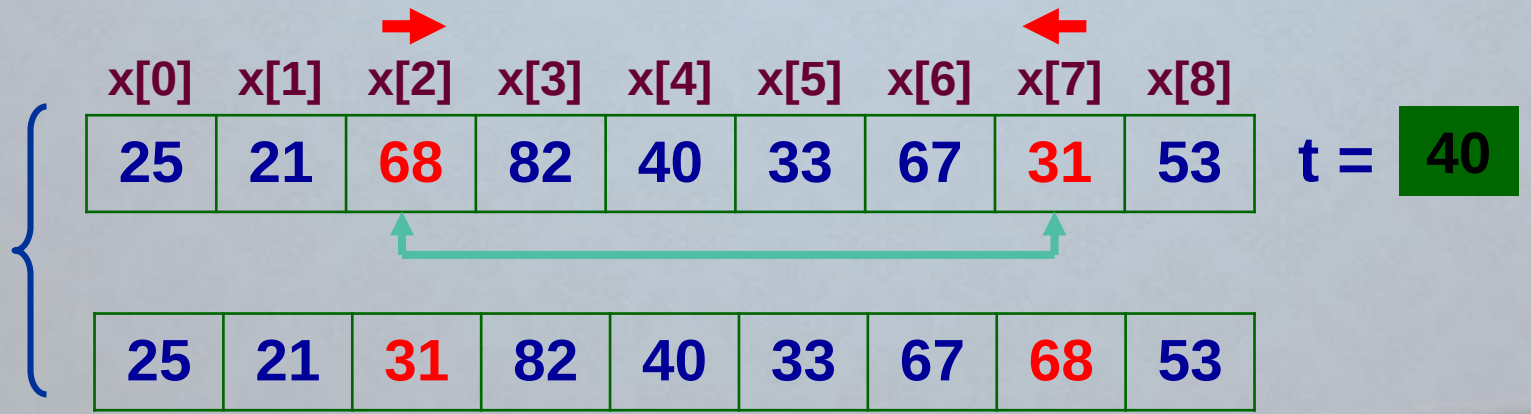
- **Thuật toán cơ sở**

```
quickSort(x[]) {
   if (n <= 1) return;
   else {
       Chọn phần tử chia t ∈ x[];
       Chia x[] thành ba dãy con
           x1[] = {e ∈ x[] | e < t}
           x2[] = {e ∈ x[] | e = t}
           x3[] = {e ∈ x[] | e > t}
       quickSort(x1[]);
       quickSort(x3[]);
   }
}
```

Sắp xếp phân đoạn (tt)

- Minh họa thuật toán trên dãy x[] có n = 9 số nguyên





Sắp xếp phân đoạn (tt)

Sắp xếp phân đoạn (tt)

- Làm tương tự với các phân đoạn X1, và X3 cho đến khi mỗi đoạn chia chỉ còn một phần tử ta có dãy X được sắp.
- Ví dụ 2: Mô tả quá trình sắp xếp dãy số sau đây theo chiều giảm dần bằng phương pháp phân đoạn.

  y = { 20 60 30 70 50 80 90 10}

Sắp xếp phân đoạn (tt)

- Thiết kế quá trình phân đoạn từ x[left] đến x[right].
  - Chọn phần tử chia t = x[(left + right) / 2].
  - Dùng hai biến chỉ số i = left và j = right.
  - i chạy sang phải, gặp x[i] không nhỏ hơn t, i dừng lại.
  - j chạy sang trái, gặp x[j] không lớn hơn t, j dừng lại.
  - Nếu i < j đổi giá trị của x[i] và x[j], i = i + 1, j = j - 1.
  - Lặp lại quá trình cho đến khi i > j.

```
void quickSort(int x[], int left, int right){
    if (left < right) {
        k = (left + right) / 2; t = x[k];
        i = left; j = right;
        do{
            while (x[i] < t) i = i + 1;
            while (x[j] > t) j = j - 1;
            if (i <= j) {
                int tg = x[i];
                x[i] = x[j]; x[j] = tg;
                i = i+1; j = j-1;
            }
        }while (i <= j);
        quickSort(x, left, j);
        quickSort(x, i, right);
    }
}
```

Sắp xếp phân đoạn (tt) – Ứng dụng

- **Cài đặt chương trình thực hiện các việc sau:**
  - **Nhập vào số nguyên nguyên dương n thỏa mãn 0<n<100.**
  - **Nhập vào một dãy n số nguyênIn dãy vừa nhập ra màn hình.**
  - **Sắp xếp dãy theo chiều tăng dần bằng thuật toán phân đoạn.**
  - **In dãy vừa sắp ra màn hình**

Sắp xếp phân đoạn (tt)

- **Bài tập 1: Cho dãy số**

  34 14 24 54 84 64 94 74 04

  - **Minh họa việc sắp xếp dãy số theo chiều tăng dần (giảm dần) bằng phương pháp phân đoạn.**
  - **Cài đặt chương trình sắp xếp dãy số.**

- **Bài tập 2: Cho dãy từ**

  John Wen Anna Henry Thor Terry Ozil Adam Dany

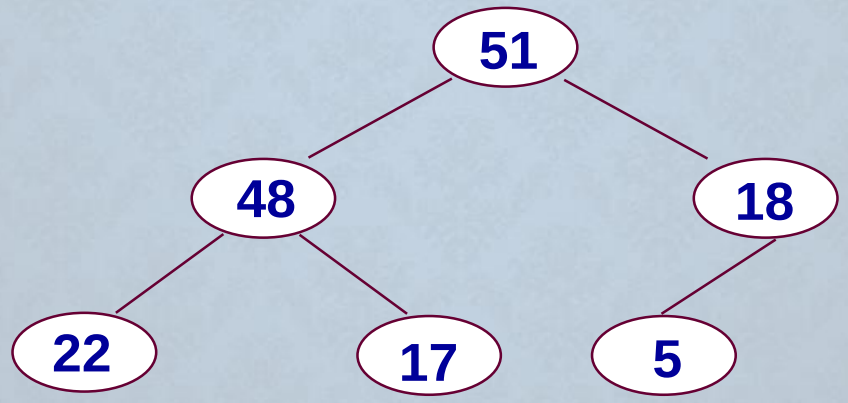
  - **Minh họa việc sắp xếp dãy từ theo trật tự từ điển (ngược lại với trật tự từ điển) bằng phương pháp phân đoạn.**
  - **Cài đặt chương trình sắp xếp dãy từ.**

Sắp xếp phân đoạn (tt)

- **Bài tập 3: Viết chương trình thực hiện các việc sau**
  - Nhập vào một danh sách học sinh (0<n<100, n nhập từ bàn phím), mỗi học sinh gồm các thông tin: Mã học sinh, họ và tên, năm sinh và điểm trung bình.
  - Sắp xếp danh sách theo chiều tăng dần của tên học sinh bằng thuật toán phân đoạn.
  - In danh sách vừa sắp ra màn hình.
  - Sắp xếp danh sách theo chiều giảm dần của điểm trung bình theo thuật toán phân đoạn.
  - In danh sách ra màn hình.

3.2.5.2. Sắp xếp vun đống – heap sort

- **Khái niệm đống - Heap**
  - Cây nhị phân trái cân đối, nút cha có giá trị lớn hơn hai con



– Cây nhị phân trái cân đối có thể lưu trong bộ nhớ bởi một mảng một chiều, theo đó nếu cha ở vị trí i thì 2 con sẽ ở các vị trí thứ 2*i + 1 và 2*i + 2.

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]
51	48	18	22	17	5

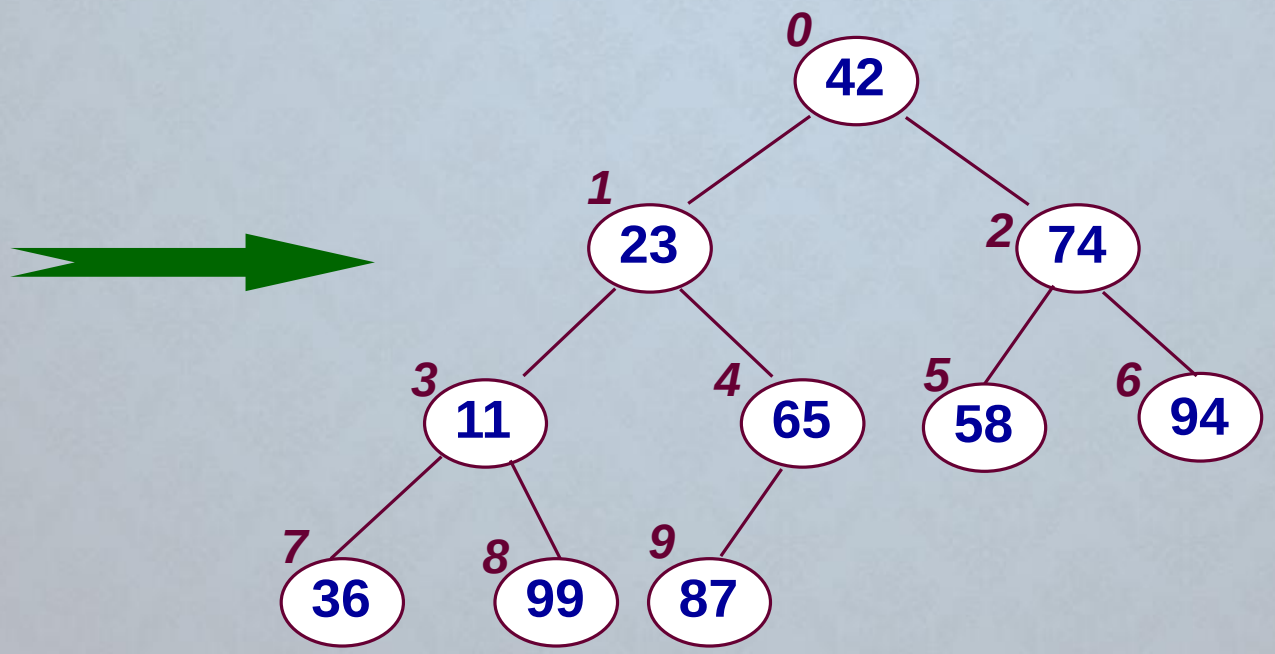
Sắp xếp vun đống (tt)

- **Nguyên tắc sắp xếp**
  - Xem dãy như cây nhị phân trái cân đối.
  - Biến đổi mảng thành cây nhị phân biểu diễn đống.
  - Đổi chỗ phần tử đầu và phần tử cuối, loại phần tử cuối.
  - Lặp lại quá trình đến khi dãy chỉ còn 1 phần tử.

Sắp xếp vun đống (tt)

- Ví dụ

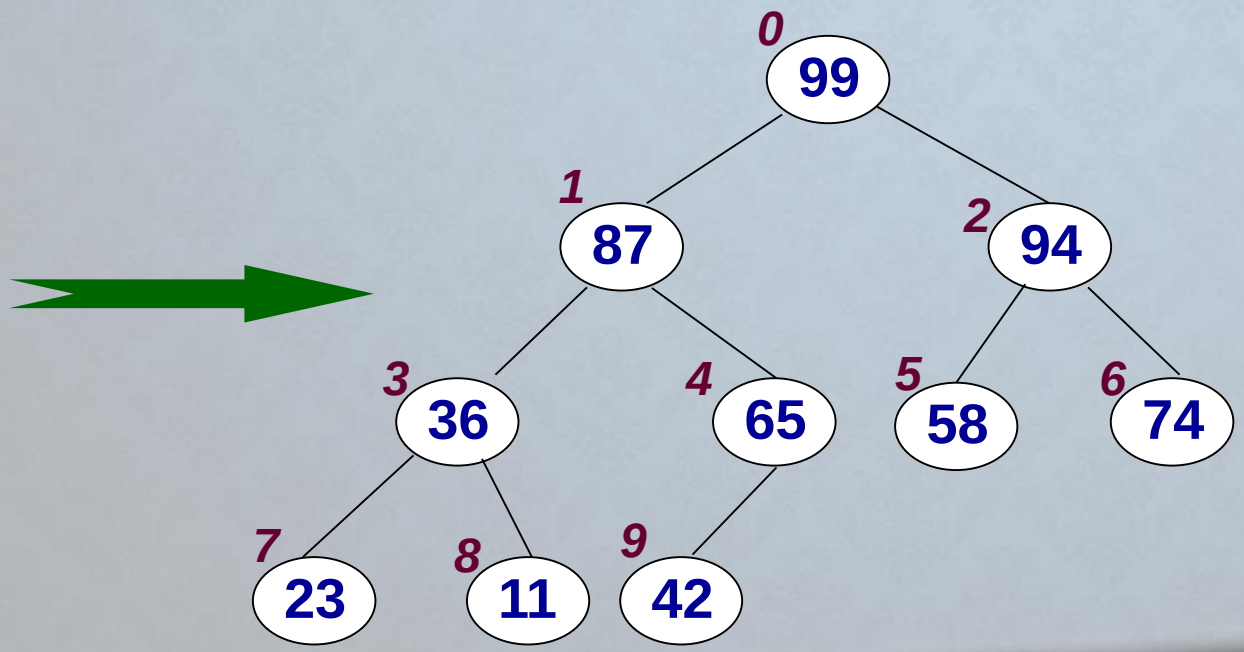
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	23	74	11	65	58	94	36	99	87



Sắp xếp vun đống (tt)

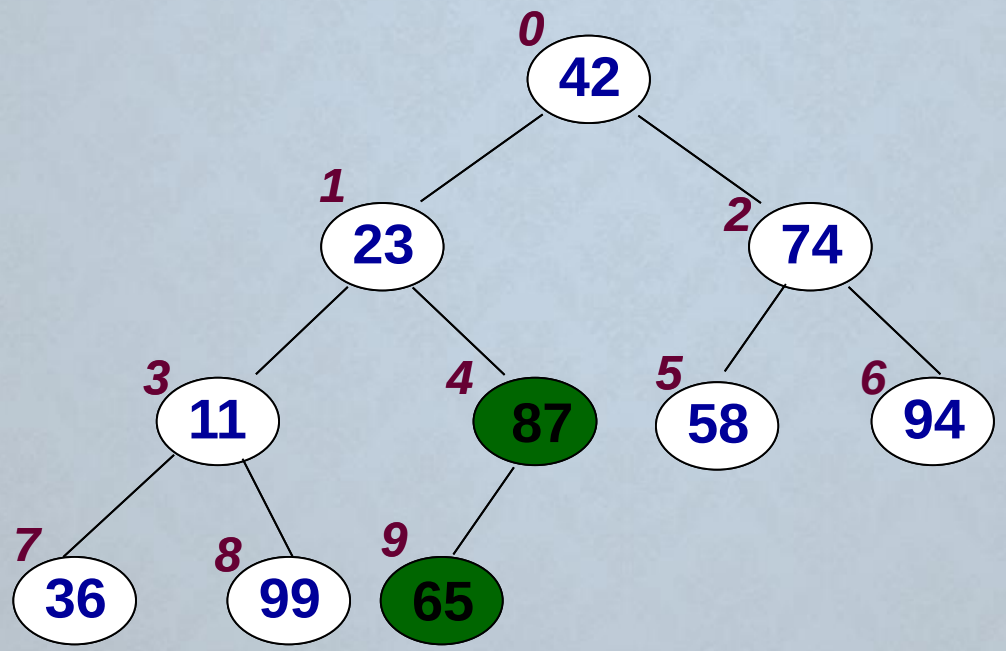
- Mảng sau khi biến đổi thành đống.

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
99	87	94	36	65	58	74	23	11	42



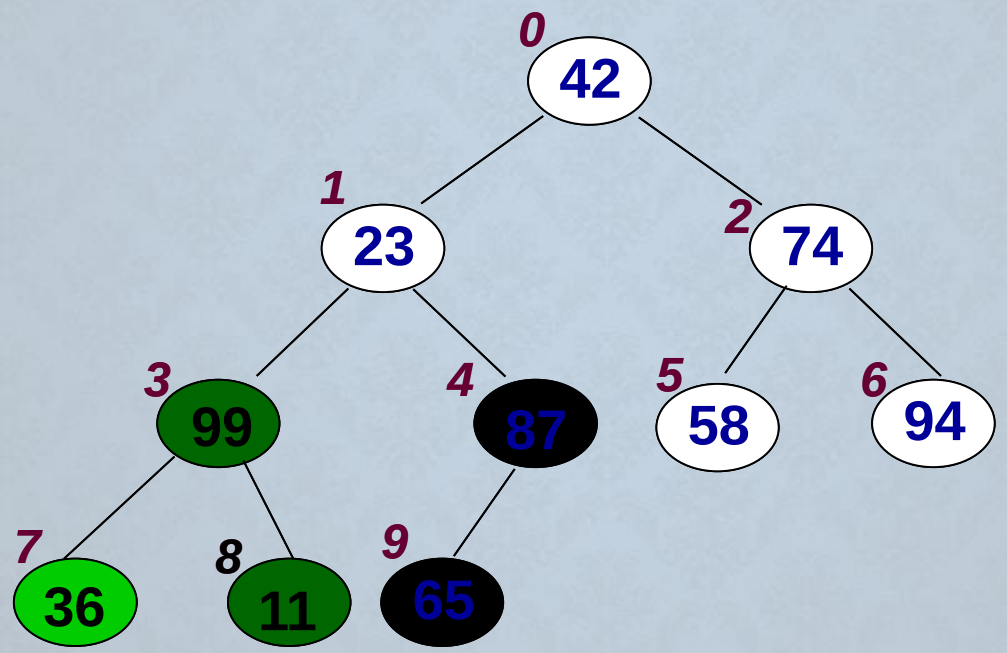
Sắp xếp vun đống (tt)

- **Quá trình biến đổi**



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	23	74	11	87	58	94	36	99	65

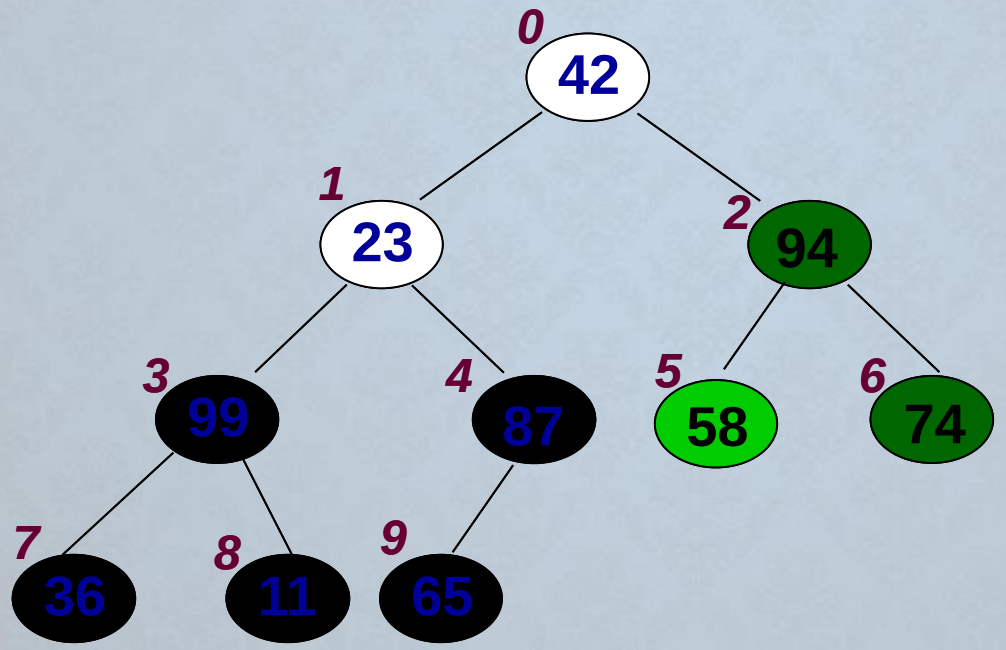
- **Quá trình biến đổi**



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	23	74	99	87	58	94	36	11	65

Sắp xếp vun đống (tt)

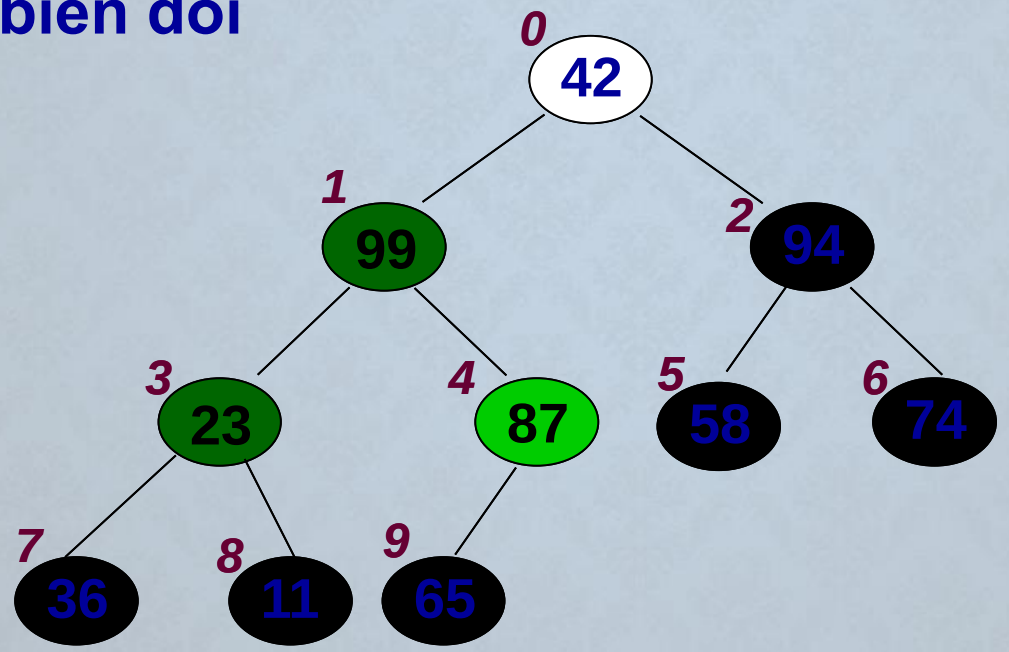
- **Quá trình biến đổi**



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	23	94	99	87	58	74	36	11	65

Sắp xếp vun đống (tt)

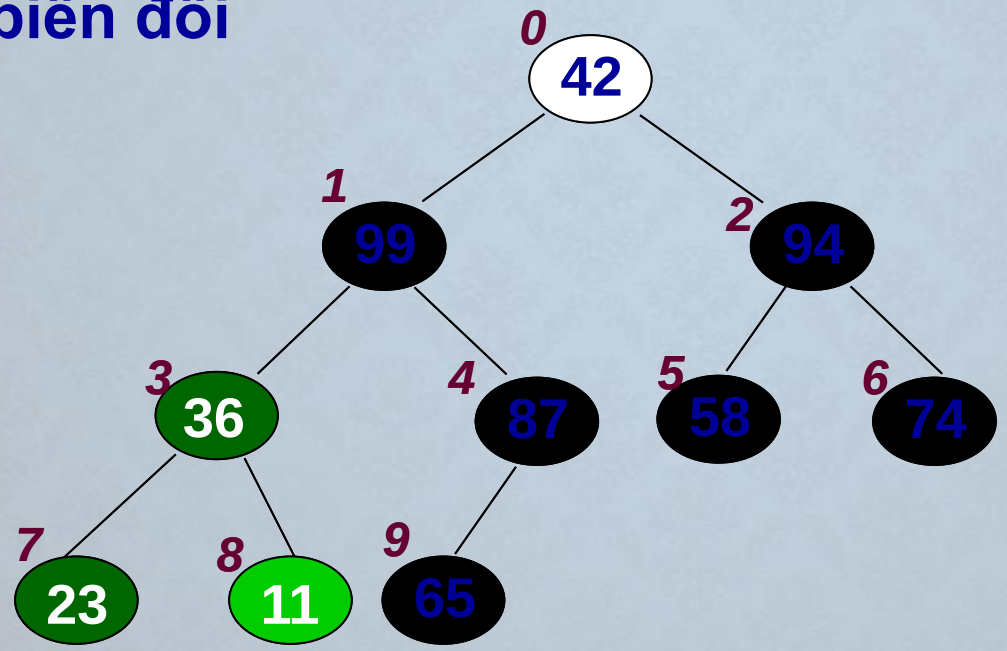
- **Quá trình biến đổi**



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	99	94	23	87	58	74	36	11	65

Sắp xếp vun đống (tt)

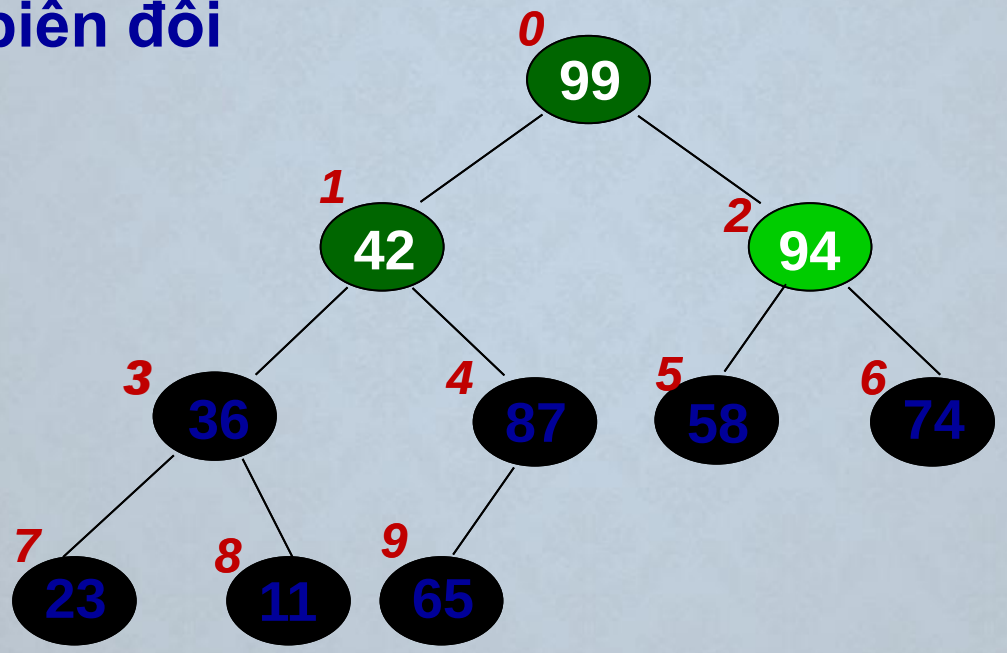
- Quá trình biến đổi



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	99	94	36	87	58	74	23	11	65

Sắp xếp vun đống (tt)

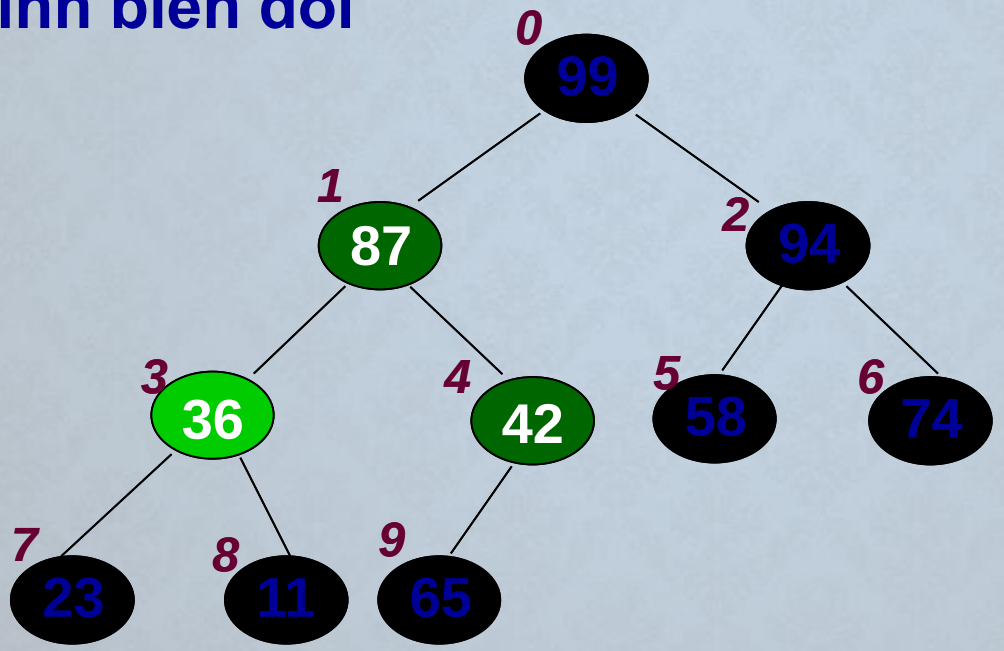
- **Quá trình biến đổi**



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
99	42	94	36	87	58	74	23	11	65

Sắp xếp vun đống (tt)

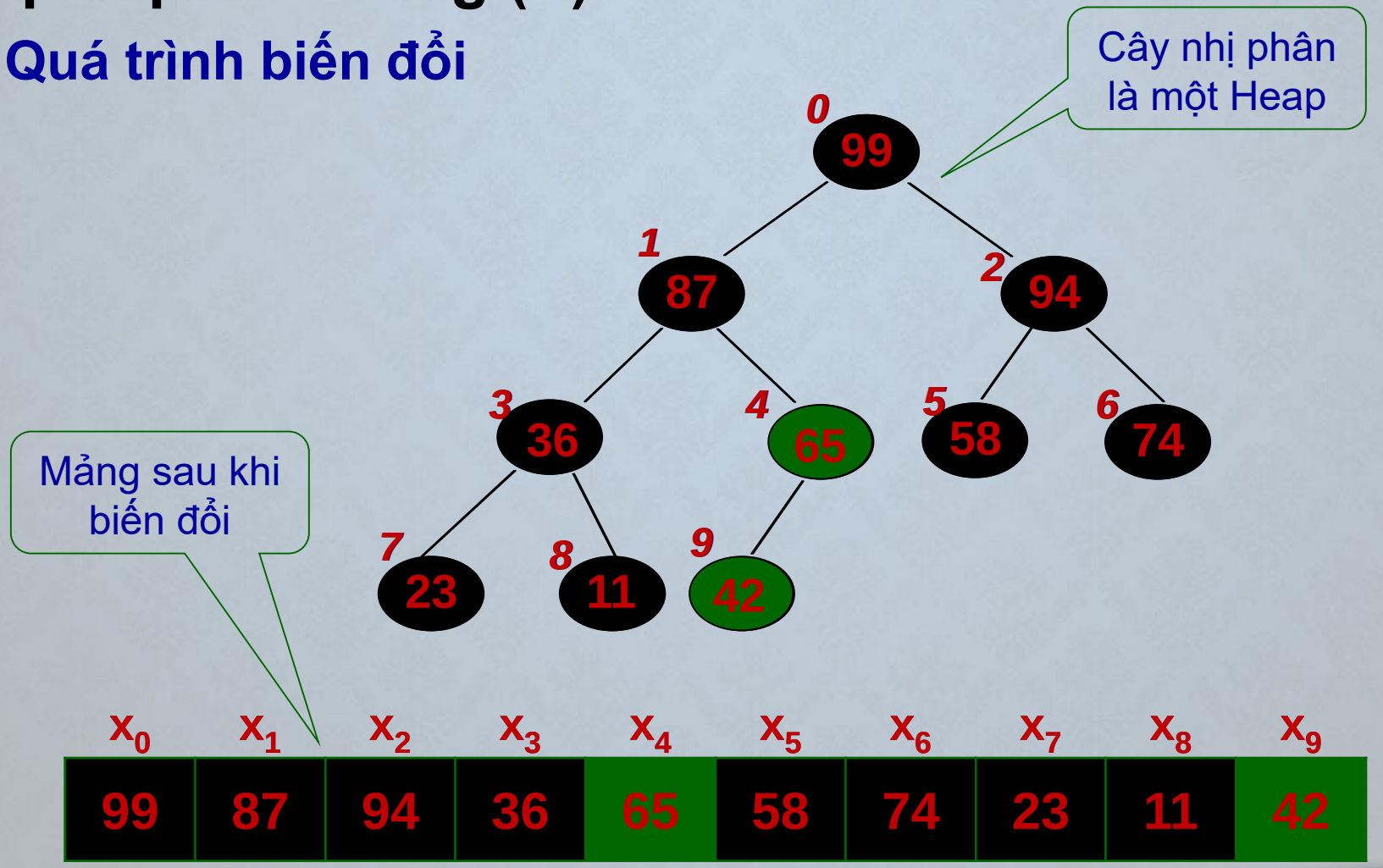
- Quá trình biến đổi



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
99	87	94	36	42	58	74	23	11	65

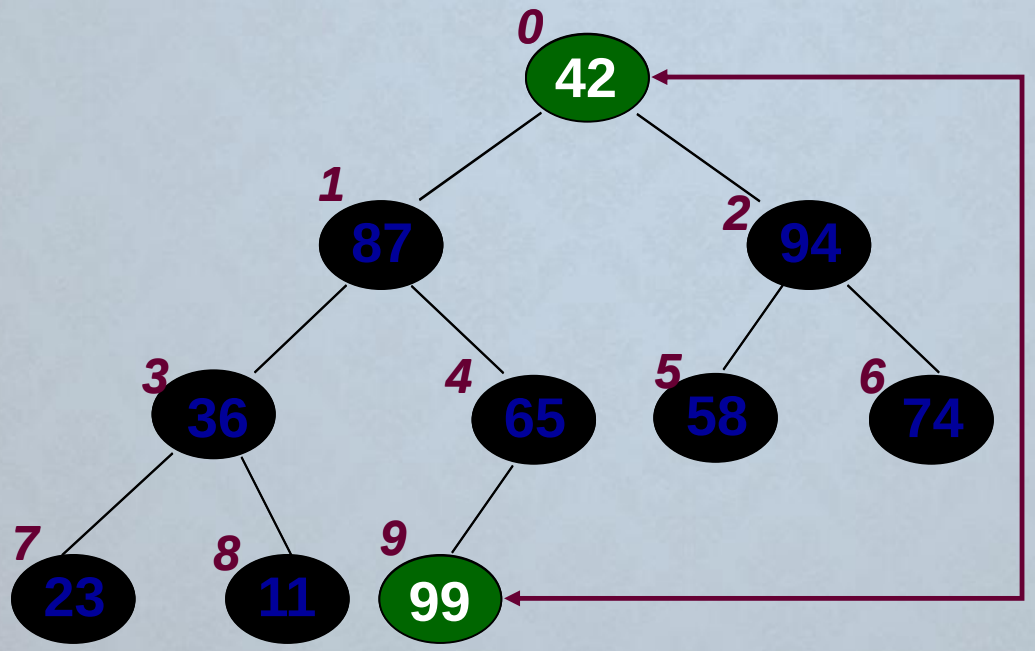
Sắp xếp vun đống (tt)

- **Quá trình biến đổi**



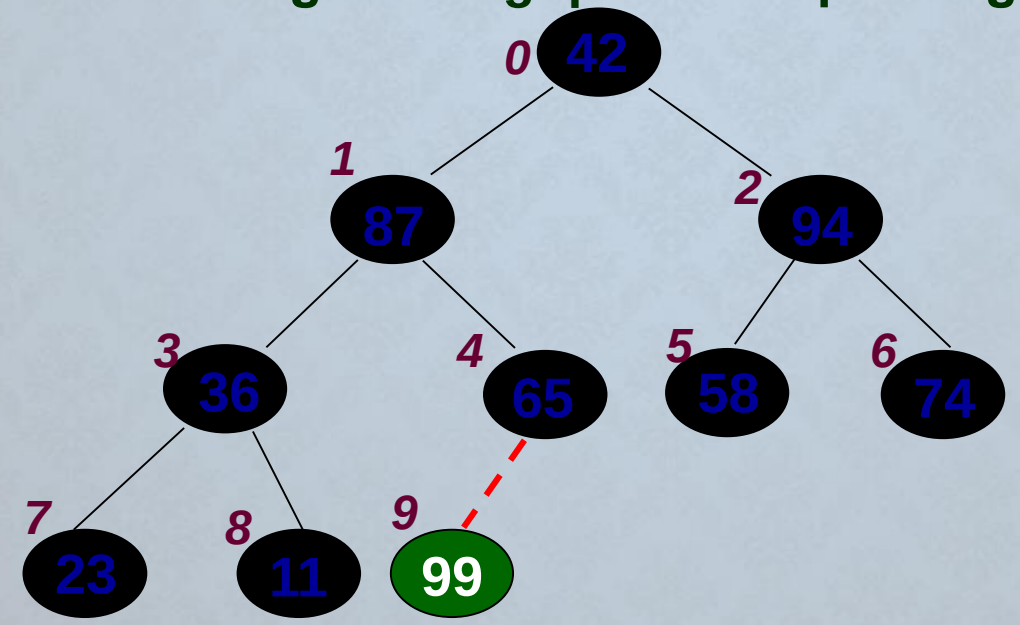
Sắp xếp vun đống (tt)

- Đổi chỗ phần tử đầu và phần tử cuối



x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	87	94	36	65	58	74	23	11	99

- **Những nhận xét:**
  - Phần tử lớn nhất ở cuối dãy, và được "loại bỏ".
  - Chỉ có nút gốc chưa là đống.
  - Từ lần 2, chỉ xét nút gốc trong quá trình tạo đống.



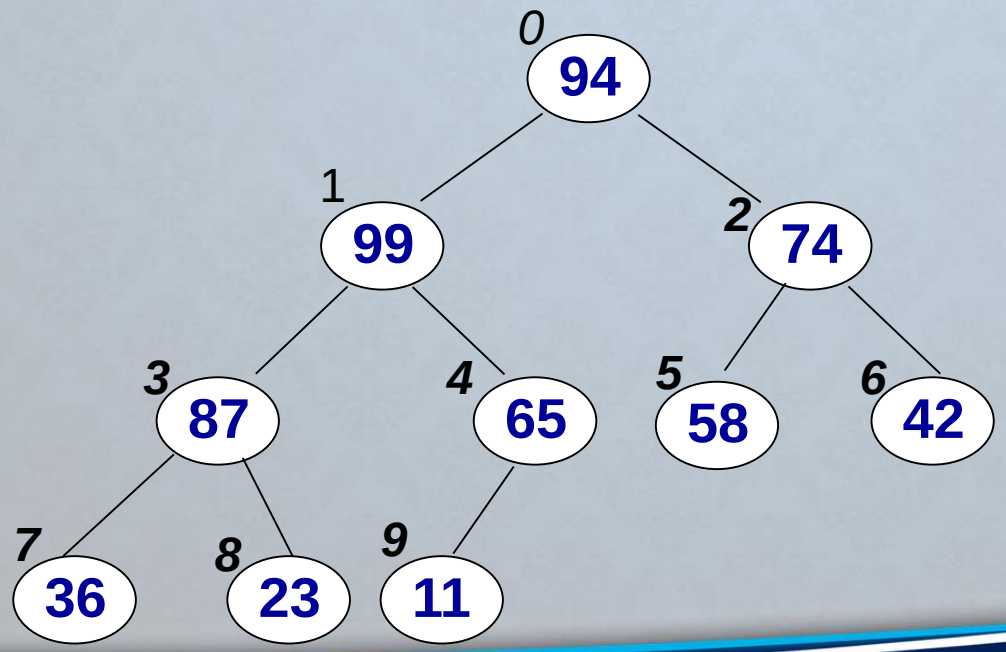
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
42	87	94	36	65	58	74	23	11	99

Sắp xếp vun đống (tt)

- Ví dụ 2: Cho dãy n số nguyên

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
94	99	74	87	65	58	42	36	23	11

- Yêu cầu: Minh họa việc sắp xếp dãy theo chiều giảm dần.



Sắp xếp vun đống (tt)

- **Thiết kế giải thuật (3 giai đoạn)**
  - **Vun đống cho một nút (1 phần tử).**
  - **Tạo thành đống đầu tiên.**
  - **Kết hợp tạo thành giải thuật heap sort.**

Sắp xếp vun đống (tt)

- **Vun đống cho một nút (1 phần tử)**
  - Nút lá là một đống
  - Vậy chỉ cần xét các nút có con.

Giải thuật vun đống cho nút x[k] trong dãy: x[0], ..., x[n-1].

```
void vunDong(x[], k, n){
    if (x[k] != lá và giá trị nhỏ hơn 2 con)
    {
        + Chọn con lớn hơn, giả sử là x[j];
        + Đổi chỗ x[k] và x[j];
        + call vunDong(x, j, n);
    }
}
```

Sắp xếp vun đống (tt)

- Vun đống cho một nút (1 phần tử)

```
void vunDong(int x[], int k, int n) {
   if (k <= n / 2 - 1) {
      int j = 2 * k + 1;
      if (j < n - 1 && x[j] < x[j + 1])
         j = j + 1;
      if (X[k] < X[j]) {
         int tg = x[k];
         x[k] = x[j];
         x[j] = tg;
         vunDong(x, j, n);
      }
   }
}
```

Sắp xếp vun đống (tt)

- **Tạo thành đống đầu tiên**
  - Chỉ có các nút từ vị trí x[n / 2 - 1] -> x[0] mới có con.
  - Với mỗi nút x[k] (k = n / 2 - 1 -> 0) vun đống cho nó.

```
void taoDongDauTien(int x[], int n)
{
      for (int k = n / 2 - 1; k >= 0; k--)
            vunDong(x, k, n);
}
```

Sắp xếp vun đống (tt)

- **Giải thuật heap sort**
  - **Tạo Đống đầu tiên**
  - **Lặp lại quá trình (*n – 1 lần*)**
    - **Đổi chỗ phần tử đầu cho phần tử cuối.**
    - **“Loại phần tử cuối”.**
    - **Vun đống cho nút đầu tiên.**

Sắp xếp vun đống (tt)

- Kết hợp tạo thành giải thuật heap sort.

```
void heapSort(int x[], int n)
{
      taoDongDauTien(x, n);
      for (int i = n; i >= 2; i--)
      {
            int tg = x[0];
            x[0] = x[i-1];
            x[i-1] = tg;
            vunDong(x, 0, i-1);
      }
}
```

Sắp xếp vun đống (tt) – Ứng dụng

- **Viết chương trình thực hiện các việc sau**
  - **Nhập vào một dãy n số nguyên (0<n<100, n nhập từ bàn phím).**
  - **In dãy vừa nhập ra màn hình.**
  - **Sắp xếp dãy theo chiều tăng dần bằng thuật toán Vun đống.**
  - **In dãy vừa sắp ra màn hình.**

Sắp xếp vun đống (tt)

- **Bài tập 1: Cho dãy số**

  34 14 24 54 84 64 94 74 04

  - **Minh họa việc sắp xếp dãy số theo chiều tăng dần (giảm dần) bằng phương pháp vun đống.**
  - **Cài đặt chương trình sắp xếp dãy số.**

- **Bài tập 2: Cho dãy từ**

  John Wenger Anna Henry Thor Terry Ozil Adam Dennis

  - **Minh họa việc sắp xếp dãy từ theo trật tự từ điển (ngược lại với trật tự từ điển) bằng phương pháp vun đống.**
  - **Cài đặt chương trình sắp xếp dãy từ.**

Sắp xếp vun đống (tt)

- **Bài tập 3: Viết chương trình thực hiện các việc sau**
  - Nhập vào một danh sách học sinh (0<n<100, n nhập từ bàn phím), mỗi học sinh gồm các thông tin: Mã học sinh, họ và tên, năm sinh và điểm trung bình.
  - Sắp xếp danh sách theo chiều tăng dần của tên học sinh bằng thuật toán vun đống.
  - In danh sách vừa sắp ra màn hình.
  - Sắp xếp danh sách theo chiều giảm dần của điểm trung bình theo thuật toán vun đống.
  - In danh sách ra màn hình.

3.2.5.3. Sắp xếp trộn – merge sort

- Tư tưởng: Trộn hai dãy đã được sắp xếp thành một dãy được sắp xếp.
- Giả sử cho hai dãy được sắp xếp theo chiều tăng dần.

  X: 12 25 28 và

  Y: 3 9 15 32 39

- Khi đó ta sẽ trộn hai dãy X và Y thành dãy Z cũng được sắp tăng như sau:

  Z: 3 9 12 15 25 28 32 39

Sắp xếp trộn (tt)

- **Mô tả tư tưởng trộn**

X: 12 25 28 và

Y: 3 9 15 32 39

→ Z: 3

X: 12 25 28 và

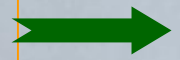
Y: 3 9 15 32 39

→ Z: 3 9

Sắp xếp trộn (tt)

X: 12 25 28 và

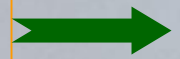
Y: 3 9 15 32 39



Z: 3 9 12

X: 12 25 28 và

Y: 3 9 15 32 39



Z: 3 9 12 15

X: 12 25 28 và

Y: 3 9 15 32 39

Z: 3 9 12 15 25

X: 12 25 28 và

Y: 3 9 15 32 39

Z: 3 9 12 15 25 28

X: 12 25 28 và

Y: 3 9 15 32 39

Z: 3 9 12 15 25 28 32 39

- **Giải thuật trộn hai dãy đã sắp xếp thành một dãy.**

```
void merging(int X[], int m, int Y[], int n, int Z[]) {
//1. Khởi tạo các chỉ số
   int i = 0, j = 0, k = 0;
//2. Chuyển các phần tử từ dãy X, Y vào dãy Z
   while (i < m && j < n) {
      if (X[i]<Y[j]) { Z[k] = X[i]; i++; k++; }
      else { Z[k] = Y[j]; j++; k++; }
   }
//3. Một dãy đã hết, đưa phần của dãy còn lại vào Z
   while (i < m) {
      Z[k] = X[i]; i++; k++;
   }
   while (j < n) {
      Z[k] = Y[j]; j++; k++;
   }
}
```

Sắp xếp trộn (tt)

- Sắp xếp bằng phương pháp trộn.
- Minh họa qua việc sắp xếp dãy số dưới đây.

x0	x1	x2	x3	x4	x5	x6	x7	x8	x9
42	23	74	11	65	58	94	36	99	87

- **Cách thực hiện:**
  - Xem dãy cần sắp gồm n dãy con nối tiếp, gọi là vệt.
  - Trộn các cặp vệt kề nhau, được vệt có độ dài gấp đôi.
  - Lặp lại quá trình trộn khi vệt có độ dài bằng dãy.

- **Minh họa phương pháp**

X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
[42]	[23]	[74]	[11]	[65]	[58]	[94]	[36]	[99]	[87]

Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9
[23	42]	[11	74]	[58	65]	[36	94]	[87	99]

X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
[11	23	42	74]	[36	58	65	94]	[87	99]

Sắp xếp trộn (tt)

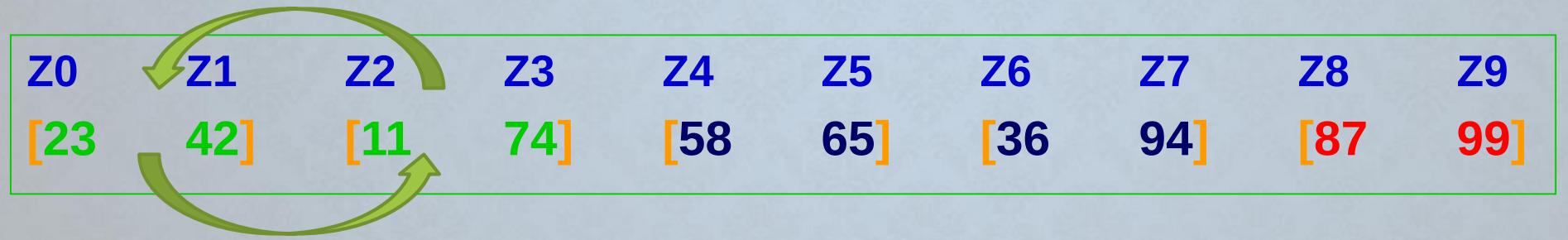
- **Ví dụ 2: Cho dãy số**

x0	x1	x2	x3	x4	x5	x6	x7	x8	x9
42	23	74	11	65	58	94	36	99	87

- **Yêu cầu: Minh họa việc sắp xếp số dãy theo chiều giảm dần.**

Sắp xếp trộn (tt)

- **Thiết kế giải thuật (3 giai đoạn)**
  - Trộn hai vệt thành một vệt.



  - Biến đổi dãy gồm các vệt độ dài K, thành dãy gồm các vệt có độ dài 2K (trộn các cặp vệt trên dãy).
  - Giải thuật trộn – merger sort.

Sắp xếp trộn (tt)

- **Thiết kế giải thuật (3 giai đoạn)**
  - Trộn hai vệt thành một vệt.
  - Biến đổi dãy gồm các vệt độ dài K, thành dãy gồm các vệt có độ dài 2K (trộn các cặp vệt trên dãy).

X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
[42]	[23]	[74]	[11]	[65]	[58]	[94]	[36]	[99]	[87]

Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9
[23	42]	[11	74]	[58	65]	[36	94]	[87	99]

  - Giải thuật trộn – merger sort.

```
void merge(int X[], int bt1, int w1, int bt2, int w2, int Z[]) {
//bt1, bt2: là vị trí biên trái của hai vệt, w1, w2 là độ dài của hai
   vệt
   int i = bt1, j = bt2, bp1 = bt1 + w1 - 1;
   int bp2 = bt2 + w2 - 1, k = bt1;
//bp1, bp2 là biên phải của hai vệt, k là biên trái của vệt mới trên Z
   while (i <= bp1 && j <= bp2) {
       if (X[i]<X[j]) { Z[k] = X[i]; i++; k++; }
       else { Z[k] = X[j]; j++; k++; }
   }
   while (i <= bp1) { Z[k] = X[i]; i++; k++; }
   while (j <= bp2) { Z[k] = X[j]; j++; k++; }
}
```

Sắp xếp trộn (tt)

- Biến đổi dãy gồm các vệt độ dài K, thành dãy gồm các vệt có độ dài 2K (trộn các cặp vệt trên dãy)
  - Trộn các cặp vệt kề nhau, thành các vệt có độ dài gấp đôi.
  - Các vệt không có cặp giữ nguyên.
- Dưới đây là thủ tục trộn các cặp vệt của dãy X, các phần tử sẽ được chuyển sang dãy Z.

```
void mergePass (int X[], int n, int K, int Z[]) {
//Z là dãy chứa dãy X sau khi trộn các cặp vệt
//1. Khởi tạo các giá trị ban đầu
   int cv = n / (2 * K); //Số cặp vệt
   int s = 2 * K * cv; //Số pt có cặp độ dài K
   int r = n - s;       //Số pt lẻ cặp
//2. Trộn từng cặp vệt
   for (int j = 1; j <= cv; j++){
      b1 = (2 * j -2) * K; //biên trái của vệt thứ nhất
      merge(X, b1, K, b1 + K, K, Z);
   }
//3. Chỉ còn một vệt
   if (r <= K)
      for (int j = 0; j < r; j++) { Z[s + j] = X[s + j]; }
//4. Còn hai vệt nhưng một vệt có độ dài nhỏ hơn K
   else merge(X, s, K, s + K, r - K, Z);
}
```

Sắp xếp trộn (tt)

- **Giải thuật sắp xếp trộn:**

```
void mergeSort (int X[], int n)
{
//1. Khởi tạo số phần tử trong một vệt
   int K = 1;
//2. Sắp xếp trộn
   while (K < n)
   {
   //Trộn và chuyển các phần tử vào dãy Z
       mergePass(X, n, K, Z);
   //Trộn và chuyển các phần tử trở lại dãy X
       mergePass(Z, n, 2 * K, X);
       K = K * 4;
   }
}
```

Sắp xếp trộn (tt) - Ứng dụng

- **Viết chương trình thực hiện các việc sau**
  - **Nhập số nguyên dương n thỏa mãn 0 < n <= 100.**
  - **Nhập vào một dãy n số nguyên.**
  - **In dãy vừa nhập ra màn hình.**
  - **Sắp xếp dãy theo chiều tăng dần bằng thuật toán sắp xếp trộn.**
  - **In dãy vừa sắp ra màn hình.**

Sắp xếp trộn(tt)

- **Bài tập 1: Cho dãy số**

  34 14 24 54 84 64 94 74 04

  - **Minh họa việc sắp xếp dãy số theo chiều tăng dần (giảm dần) bằng phương pháp trộn.**
  - **Cài đặt chương trình sắp xếp dãy số.**

- **Bài tập 2: Cho dãy từ**

  John Wenger Anna Henry Thor Terry Ozil Adam Dennis

  - **Minh họa việc sắp xếp dãy từ theo trật tự từ điển (ngược lại với trật tự từ điển) bằng phương pháp trộn.**
  - **Cài đặt chương trình sắp xếp dãy từ.**

Sắp xếp trộn(tt)

- **Bài tập 3: Viết chương trình thực hiện các việc sau**
  - Nhập vào một danh sách học sinh (0<n<100, n nhập từ bàn phím), mỗi học sinh gồm các thông tin: Mã học sinh, họ và tên, năm sinh và điểm trung bình.
  - Sắp xếp danh sách theo chiều tăng dần của tên học sinh bằng thuật toán trộn.
  - In danh sách vừa sắp ra màn hình.
  - Sắp xếp danh sách theo chiều giảm dần của điểm trung bình theo thuật toán trộn.
  - In danh sách ra màn hình.