THUẬT TOÁN

Các thuật toán sắp xếp

21/09/2015 6,549 Views
13 Min Read

Xin chào mọi người! Hôm nay quởn đời, lại không có gì làm, lục lại cuốn sách cũ và phát hiện thấy mấy bài toán cũng vui vui, nhìn quanh quanh thấy lại mấy bài toán sắp xếp nên quyết định làm 1 bài tổng hợp về các thuật toán sắp xếp.

Với tất cả các loại sort, thì mình sẽ sử dụng hàm **swap** (hoán vị) để thuận tiện cho việc thao tác nhé!

Mã nguồn:

```
1  void swap(int&a, int&b)
2  {
3    int temp = a;
4    a = b;
5    b = temp;
6  }
```

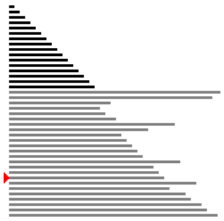
Và mình quy ước: **a** là tên mảng, **n** là số phần tử 😀

Thời gian chạy của các kiểu sort

Thôi thì mình cùng xem những thuật toán từ đơn giản đến phức tạp nhé!

1. Bubble sort

Bubble sort (hay còn gọi là Sắp xếp nổi bọt) là 1 thuật toán dễ cài đặt nhất trong tất cả các loại thuật toán, vì vậy nó được mấy bạn học sinh, sinh viên ưa xài (mình lâu lâu làm biếng cũng hay xài :v). Như tên gọi, thuật toán này sẽ tìm vị trí được xem là nhỏ nhất trong dãy bằng cách bắt cặp so sánh, rồi cho nó "trồi" lên vị trí đầu tiên, tiếp tục tới những phần tử tiếp theo, nó cứ trồi lên từ bé đến lớn, giống như những bong bóng nổi bọt vậy.



Mã nguồn:

```
1  void BubbleSort(int a[], in
2  {
3    for (int i = 0; i < n - 1
4    {
5      for (int j = n - 1; j >
6      {
7        if (a[j] < a[j - 1])
8      }
9    }
10 }
```

Vì đây là thuật toán dễ cài đặt, sử dụng nên nó cũng khá là chậm. Với trường hợp xấu nhất có thể xảy ra, độ phức tạp của thuật toán này là $O(n^2)$.

2. Insertion sort

Insertion sort (hay còn gọi là Sắp xếp chèn) là 1 thuật toán khá đơn giản bằng cách chọn những phần tử nhỏ nhất để đưa lên đầu trong 1 dãy. Giống như ta đánh bài "tiến lên miền Nam" (mình không có tuyên truyền cờ bạc nha :v), chọn 13 lá. Duyệt từ trái qua phải, thấy lá nào nhỏ hơn 1 dãy lá bài bên trái thì bốc ra và nhét vào đúng vị trí sao cho dãy đã sắp xếp luôn tăng/giảm dần.

Insertion sort cũng vậy, ban đầu ta có 1 vị trí gọi là **x**. Dãy các phần tử từ **0 -> (x – 1)** là dãy đã sắp xếp, còn dãy từ **x -> n** là dãy chưa được sắp xếp. Nhiệm vụ của ta là duyệt dãy **x -> n**, thấy phần tử nào thì tìm vị trí phù hợp trong dãy **0 -> (x – 1)** mà nhét vào.

Mã nguồn:

```
1  void InsertionSort(int a[],
2  {
3    int x, temp;
4    for (int i = 1; i < n; i+
5    {
6      x = i - 1; // Bắt đầu t
7      temp = a[i]; // Duyệt n
8      while (x >= 0 && a[x] >
9      {
10       a[x + 1] = a[x];
11       x--;
12     }
13     a[x + 1] = temp; // Gán
14   }
15 }
```

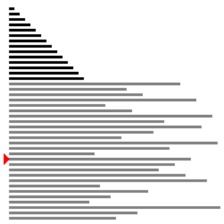
Insertion sort trong trường hợp tốt nhất (tất cả các phần tử đã được sắp xếp sẵn) thì độ phức tạp của thuật toán là O(n) (vì chỉ cần duyệt n lần từ trái qua, không có chèn). Tuy nhiên, với trường hợp xấu nhất thì độ phức tạp có thể lên tới $O(n^2)$.

3. Selection sort

Selection sort (hay còn gọi là Sắp xếp chọn), là 1 trong những thuật toán dễ cài đặt và học nhất. Đúng như tên gọi,

ta chỉ cần duyệt hết mảng, thấy thằng nào nhỏ nhất thì nắm đầu nó vứt lên đầu bảng.

Để làm được điều này, ta cần có 1 biến **x** tương tự như *Insertion sort*. Ta duyệt từ **x -> n**, nếu tìm được phần tử nhỏ nhất thì ta hoán vị nó với vị trí **x**.



Mã nguồn:

```
1  void SelectSort(int a[], int
2  {
3    for (int i = 0; i < n; i++
4    {
5      int x = i;
6      for (int j = x; j < n; j
7      swap(a[i], a[x]);
8    }
9  }
```

Thuật toán này mặc dù có ưu việt là ít đổi chỗ các phần tử nhất, tuy nhiên vì ta phải tìm **x**, nên độ phức tạp của thuật toán cũng tương đương là $O(n^2)$.

4. Merge sort

Merge sort (hay còn gọi là Sắp xếp trộn) là 1 thuật toán điển hình của lối thuật toán **Chia để trị** (Divide and Conquer, D&C). Về ý tưởng thì ta chia 1 mảng lớn thành 2 mảng con, đến 1 điều kiện cụ thể (thường là chỉ còn 2 phần tử) rồi so sánh, sau đó gom từng mảng con lại thành mảng lớn để ra kết quả.

Giả sử ta có 2 mảng đã được sắp xếp sẵn là A = { 1, 3, 6, 9 } và B = { 2, 4, 5 }, ta muốn trộn 2 mảng đó thành 1 mảng lớn là AB thì ta cần phải so sánh 2 vị trí đầu tiên, nếu vị trí nào bé/lớn hơn thì ta cho vào mảng trộn:

- A = { 1, 3, 6, 9 }, B = { 2, 4, 5 }, 1 < 2 => AB = { 1 }
- A = { 3, 6, 9 }, B = { 2, 4, 5 }, 3 > 2 => AB = { 1, 2 }

- A = { 3, 6, 9 }, B = { 4, 5 }, 3 < 4 => AB = { 1, 2, 3 }
- A = { 6, 9 }, B = { 4, 5 }, 6 > 4 => AB = { 1, 2, 3, 4 }
- A = { 6, 9 }, B = { 5 }, 6 > 5 => AB = { 1, 2, 3, 4, 5 }
- A = { 6, 9 }, B = { }, B rỗng => nhét toàn bộ mảng A vào AB = { 1, 2, 3, 4, 5, 6, 9 }

Mã nguồn:

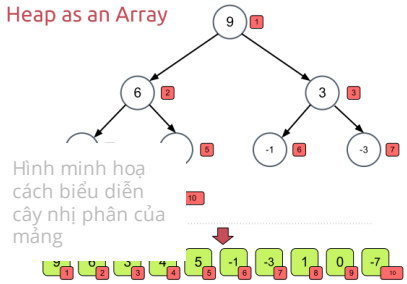
```
void Merge(int a[], int lef
{
  // Tạo 2 vị trí mảng con
  int left1 = left, right1
  int index = left;
  int *b = new int[right -

  while (left1 <= right1 &&
  {
    if (a[left1] < a[left2]
    {
      b[index] = a[left1];
      index++; left1++; //
    }
    else
    {
      // Tương tự trên
      b[index] = a[left2];
      index++; left2++;
    }
  }
  // Trường hợp 1 trong 2 m
  if (left2 > right2)
  {
    while (left1 <= right1)
    {
      b[index] = a[left1];
      index++; right1++;
    }
  }
  if (left1 > right1)
  {
    while (left2 <= right2)
    {
      b[index] = a[left2];
      index++;
      left2++;
    }
  }

  // Cuối cùng, ta gán mảng
  for (index = left; index
  {
    a[index] = b[index];
  }
}

```

```
48  void MergeSort(int a[], int
49  {
50    if (left < right)
51    {
52      int mid = (left + right
53      // Bắt đầu chia mảng ló
54      MergeSort(a, left, mid)
55      MergeSort(a, mid + 1, r
56      // Chia xong thì trộn t
57      Merge(a, left, mid, rig
58    }
59  }
```

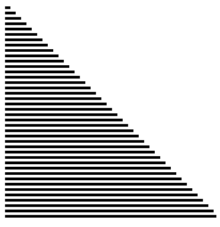
Tất nhiên, Merge sort có ưu điểm là nhanh hơn hẳn 3 loại sort phía trên, độ phức tạp của thuật toán O(n*log(n)). Tuy nhiên bạn phải hiểu rõ cơ chế của thuật toán, cộng với việc cài đặt tương đối khó.

5. Heap sort

Heap sort (hay còn gọi là Sắp xếp vun đống) là dạng sắp xếp dựa trên **cây nhị phân**. Cây nhị phân được xác định như sau: tại nút i có nút con trái là 2*(i + 1) – 1 và nút con phải là 2*(i + 1).



Ta xây dựng heap sao cho nút cha đều lớn hơn 2 nút con. Khi đó nút gốc có giá trị lớn nhất. Sau đó ta hoán vị nút gốc với nút thứ **n – 1** và xây dựng lại heap mới từ **0 -> (n – 2)** và nút cha lớn hơn 2 nút con. Lặp đi lặp lại việc xây dựng và hoán vị nút gốc với nút cuối trong heap, ta sẽ được mảng đã sắp xếp.



Mã nguồn:

```
1  // Ta xem i là nút cha
```

```
2  void MaxHeapify(int a[], in
3  {
4    int left = 2*(i + 1) - 1;
5    int right = 2*(i + 1);
6    int max;
7
8    // Tìm nút cha lớn nhất
9    if (left < n && a[left] >
10   else max = i;
11   if (right < n && a[right]
12
13   if (i != max)
14   {
15     swap(a[i], a[max]);
16     MaxHeapify(a, n, max);
17   }
18 }
19 void BuildHeap(int a[], int
20 {
21   // Xây dựng heap chỉ lặp
22   // Do ta cần xây dựng cây
23   // Vì vậy mảng cần phải d
24   for (int i = n / 2 - 1; i
25   {
26     Heapify(a, n, i);
27   }
28 }
29 void HeapSort(int a[], int
30 {
31   BuildHeap(a, n);
32   // Lúc này a[0] là phần t
33   for (int i = n - 1; i > 0
34   {
35     swap(a[0], a[i]);
36     Heapify(a, i, 0);
37   }
38 }
```

Tương tự Merge sort, Heap sort có độ phức tạp của thuật toán O(n*log(n)). Vì vậy nên việc cài đặt và sử dụng Heap sort tương đối khó, và cần phải hiểu về cây nhị phân mới có thể sử dụng tốt.

6. Quicksort

Quicksort (còn được gọi là Sắp xếp nhanh) là 1 dạng thuật toán **Divide and Conquer** giống như Merge sort. Theo ý kiến cá nhân của mình, thì Quicksort được xem là thuật toán sắp xếp nhanh nhất (vậy nên nó mới có chữ "quick" chứ :v).

Quicksort chọn 1 phần tử trong mảng là **chốt** (pivot), thường ở vị trí giữa cho dễ tính (ta có thể chọn ở bất kì đâu, nhưng để tránh rơi vào vòng lặp vô tận thì cứ chọn ở giữa đi 😀). Ta đưa những phần tử nhỏ hơn pivot về danh sách thứ 1, và những phần tử lớn hơn pivot về danh sách thứ 2. Cứ như vậy,

ta dùng đệ quy ở 2 phần danh sách con đến khi nào chỉ còn 1 phần tử.

Mã nguồn:

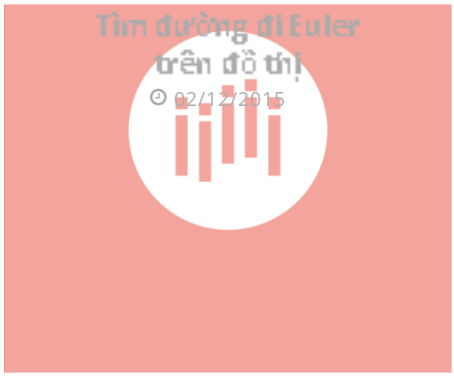
```
void QuickSort(int a[], int
{
  int i = left, j = right;
  int pivot = a[(left + rig
  do
  {
    // Tìm vị trí i, j cần
    while (a[i] < pivot &&
    while (a[j] > pivot &&
    if (i <= j)
    {
      swap(a[i], a[j]);
      i++; j--;
    }
  } while (i <= j);

  // Khi đó pivot sẽ chốt v
  // Ta cần gọi đệ hàm đến
  if (left < j) QuickSort(a
  if (i < right) QuickSort(
}
```

Quicksort cũng tương tự như Merge sort, cũng có độ phức tạp thuật toán O(n*log(n)). Tuy nhiên, việc cài đặt Quicksort ngắn gọn hơn hẳn so với Merge sort hay Heap sort, và mình cũng rất hay sử dụng Quicksort.

Trên đây, mình đã giới thiệu 6 kiểu sort phổ biến, còn rất nhiều kiểu sort khác nhưng do blog đã dài, và mình cũng chưa nghiên cứu hết nên không đề cập ở đây. Cảm ơn các bạn đã chú ý theo dõi, hẹn gặp lại ở những bài tiếp theo!

Tags thuat_toan

You may also like



About the author VIEW ALL POSTS

Võ Hoài Sơn

Tính tình bất định
Chọc vào là bịnh
Rất yêu lập trình
Luôn code hết mình
Mình hiện đang là sinh viên của trường ĐH Khoa học tự nhiên TPHCM. Bản thân rất thích code, kiêm luôn cả mần thơ nên thường hơi hâm hâm dở dở. Ngoài ra chém gió, chém chuối, chém trái cây các kiểu cũng là sở trường của mình. Rất mong được làm quen với các bạn :D