Giải Thuật Lập Trình

Nơi tổng hợp và chia sẻ những kiến thức liên quan tới giải thuật nói chung và lý thuyết khoa học máy tính nói riêng.

‹ Khái niệm tiệm cận -- Asymptotic Notation • Sắp xếp-- quicksort and merge sort ›

Chọn phần tử lớn thứ k của mảng--Median Selection

May 28, 2015 in Uncategorized | No comments

Trong bài này chúng ta giải bài toán sau:

Problem (k-th selection): Cho mảng $A[1,2,\ldots,n]$ gồm n phần tử. Tìm phần tử lớn thứ k của mảng.

Có hai cách giải mà chúng ta có thể nghĩ đến ngay:

1. Dựa vào định nghĩa: phần tử thứ k lớn hơn (hoặc bằng) $k-1$ phần tử của mảng và nhỏ hơn (hoặc bằng) $n-k$ phần tử khác. Như vậy ta có thể giải với thời gian $O(n^2)$ bằng cách với mỗi phần tử, kiểm tra tính chất trên trong thời gian $O(n)$.
2. Sắp xếp mảng theo chiều tăng dần và lấy ra phần tử thứ k của mảng đã sắp xếp. Sắp xếp (http://www.giaithuatlaptrinh.com/?p=41) có thể thực hiện trong thời gian $O(n\log n)$. Như vậy bài toán trên có thể giải trong thời gian $O(n\log n)$.

Bài này giới thiệu phương pháp **chia để trị** để giải bài toán trên trong thời gian $O(n)$.

Lemma: Tồn tại thuật toán chọn ra phần tử lớn thứ k của mảng $A[1,2,\ldots,n]$ trong thời gian $T(n) = O(n)$.

Ý tưởng cơ bản

Ý tưởng cơ bản của thuật toán như sau: Chọn một phần tử (bất kì), $A[p]$, và chia mảng ra thành hai phần (trong thời gian $O(n)$): phần 1 gồm những phần tử nhỏ hơn $A[p]$ và phần 2 gồm những phần tử lớn hơn (hoặc bằng) $A[p]$. Ta có hai trường hợp:

1. Nếu k **nhỏ hơn** kích thước của phần 1, chúng ta đệ quy trên phần 1 để tìm phần tử thứ k.
2. Nếu k **lớn hơn** kích thước của phần 1, chúng ta đệ quy trên phần 2 tìm phần tử thứ $r-k$, ở đây r là kích thước của phần 1.

Giả mã của thuật toán chia mảng với phân tử chọn (pivot) là $A[p]$, trả lại vị trí mới cuả phần tử có giá trị $A[p]$ trong mảng.

PARTITION$(A[1,2,\ldots,n],p)$:
    swap $A[p] \leftrightarrow A[n]$
    $i \leftarrow 0$
    $i \leftarrow n$
    while $i < j$
        repeat $i \leftarrow i+1$ until $(i \geq i \text{ or } A[i] \geq A[n])$
        repeat $i \leftarrow j-1$ until $(i \geq i \text{ or } A[j] \geq A[n])$
        if $i < j$
            swap $A[i] \leftrightarrow A[j]$
    swap $A[i] \leftrightarrow A[n]$
    return i

Dưới đây là code C của giả mã trên.

+ expand source (#)

Như vậy ta đã thực hiện xong bước một: phân chia mảng bằng phần tử bất kì. Sau đây là giả mã thực hiện bước đệ quy:

QUICKSELECT$(A[1,2,\ldots,n],k)$:
    if $n == 1$
        return $A[1]$
    else
        Choose a pivot $A[p]$
        $r \leftarrow$ PARTITION$(A[1,2,\ldots,n],p)$
        if $k < r$
            return QUICKSELECT$(A[1,2,\ldots,r-1],k)$
        else if $k > r$
            return QUICKSELECT$(A[r+1,2,\ldots,n],k-r)$
        else
            return $A[r]$

Dưới đây là code C của giả mã trên. Code đầy đủ với phần tử pivot là phần tử giữa của mảng được cho ở cuối bài viết.

```
// x, y is the first and last index of array arr
// k is the index of the choosen element
int quick_select(int arr[], int x, int y, int k){
    if(y <= x) return arr[x];
    else {
           int p = (y+x)/2; // choose the mid element to be the 
        int r = partition(arr, x, y, p);
        if ( k < r){
            return quick_select(arr, x, r-1, k);
        }else if (k > r) {
            return quick_select(arr, r+1, y, k);
        }else {
            return arr[r];
        }
    }
}
}
```

Chọn pivot tốt

Nhìn vào giả mã (#rec-code) ta dễ thấy thời gian chạy trong trường hợp tồi nhất như sau:

$$\begin{aligned} T(n) &= \max_{1\leq r\leq n}(\max(T(r-1), T(n-r)) + O(n)) \\ &= \max_{0\leq \ell \leq n-1} T(\ell) + O(n) \end{aligned}$$

Trường hợp xấu nhất khi $\ell = n-1$, khi đó $T(n) = O(n^2)$, không tốt hơn thuật toán vét cạn. Tuy nhiên, nếu ta chọn pivot sao cho $\ell \leq \alpha n$ với $\alpha < 1$, ta sẽ có

$$T(n) \leq T(\alpha n) + O(n)$$

Giải công thức truy hồi (http://www.giaithuatlaptrinh.com/?p=22) trên ta được $T(n) = O(n)$. Trong những trường hợp không biết chọn phần tử nào là pivot tốt, cách đầu tiên có thể nghĩ tời là chọn **ngẫu nhiên** một phần tử của mảng làm pivot với xác suất như nhau. Với bài toán này, chọn ngẫu nhiên sẽ cho ta thuật toán với thời gian **kì vọng (expected)** là $O(n)$. Chi tiết phân tích bạn đọc có thể xem tại đây (http://www.giaithuatlaptrinh.com/?p=950).

Blum, Floyd, Pratt, Rivest và Tarjan [2] đề xuất một cách khác để chọn pivot như sau: nếu số phần tử của mảng là lớn (≥ 25 trong giả mã), ta chia mảng thành $\lceil n/5 \rceil$ nhóm, mỗi nhóm gồm 5 phần tử. Dùng thuật toán vét cạn để tìm median của mỗi nhóm. Sau bước đó, ta thu được mảng gồm $\lceil n/5 \rceil$ phần tử là median của $\lceil n/5 \rceil$ nhóm. Gọi đệ quy để tìm median của mảng này. Lấy median làm pivot cho mảng ban đầu. Giả mã như sau:

```
LinearSelection(A[1,2,...,n], k):
    if n ≤ 25
        use brute force
    else
        m ← ⌈n/5⌉
        for i ← 1 to m do
            M[i] ← MedianOfFive(A[5i−4,...,5i])
        A[p] ← LinearSelection(M[1,2,...,m], ⌊m/2⌋)
        r ← Partition(A[1,2,...,p], ⌊m/2⌋)
        if k < r
            return LinearSelection(A[1,2,...,r−1], k)
        else if k > r
            return LinearSelection(A[r+1,2,...,n], k−r)
        else
            return A[r]
```

Dưới đây là code C của giả mã trên. Code đầy đủ được cho ở cuối bài viết.

```
int linear_selection(int _array[], int x, int y, int k){
    if(y - x <= 24) {
    return med_exhaustive(_array, x, y ,k);//brute force search
     } else {
    int m = (y-x+1)/5 + ((y-x+1)%5 > 0? 1 : 0); // m = ceiling(n
    int _brray[m];
    int i = 0;
        for (i = 0 ; i < m-1 ; i++){
           _brray[i] = med_of_five(_array[5*i+x], _array[5*i+1+x
        }
        _brray[m-1] = _array[y];
        int med_of_brray = linear_selection(_brray, 0, m-1, m/2-
        int p  =  0;
```

```
        for( i = x ; i <= y  ; i++){
           if (_array[i] == med_of_brray) p = i;
        }
        int r = partition(_array, x, y, p);
    if ( k < r){
        return linear_selection(_array, x, r-1, k);
    }else if (k > r) {
        return linear_selection(_array, r+1, y, k);
    }else {
        return _array[r];
    }
    }
  }
```

Phân tích thời gian

Ta thấy tìm median của medians mất thời gian $T(n/5)$. Vì phần tử pivot là median của median, mỗi bước đệ quy ta sẽ bỏ bớt được ít nhất là $3n/10$ phần tử (**tại sao?**). Như vậy tổng thời gian của thuật toán là:

$$\boxed{T(n) = T(n/5) + T(7n/10) + O(n)}$$

Giải công thức đệ quy (http://www.giaithuatlaptrinh.com/?p=22) trên, ta được $T(n) = O(n)$.

Dưới đây là code và dữ liệu test: Code (http://www.giaithuatlaptrinh.com/wp-content/uploads/2015/05/code-med-selection.tar.gz), Data (http://www.giaithuatlaptrinh.com/wp-content/uploads/2015/05/code-med-selection.tar.gz)

Tham khảo

Bài viết dự chủ yêu trên notes của Jeff Erickson http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/01-recursion.pdf (http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/01-recursion.pdf)

Tài liệu tham khảo liên quan:

[1] Avrim Blum: Algorithm Lecture Notes (http://www.cs.cmu.edu/~avrim/451f11/lectures/lect0908.pdf), CMU, 2011.
[2] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. *Time Bounds for Selection*. Journal of Computer and System Sciences, 7(4), 448-461. 1973.

(http://www.giaithuatlaptrinh.com/?p=35&share=twitter&nb=1)

(http://www.giaithuatlaptrinh.com/?p=35&share=facebook&nb=1)

(http://www.giaithuatlaptrinh.com/?p=35&share=google-plus-1&nb=1)

RELATED

Sắp xếp-- quicksort and merge sort (http://www.giaithua… p=41)
May 28, 2015
In "divide and conquer"

Chọn thế nào là tốt? -- How to avoid bad cases? (http://www.giaithua… p=950)
March 23, 2016
In "chocolate-bar-breaking"

Mảng hậu tố-- Suffix Array (http://www.giaithua… p=488)
September 6, 2015
In "Rabin Karp"

Tags: divide and conquer, median selection, recursion

No comments

Comments feed for this article

Trackback link: http://www.giaithuatlaptrinh.com/wp-trackback.php?p=35

Reply

Your email address will not be published. Required fields are marked *

Your comment

Name *

Email *

Website

Post Comment

Notify me of follow-up comments by email.

Notify me of new posts by email.