Phân Tích Thuật Toán

Bùi Thị Thanh Phương

Ngày 25 tháng 4 năm 2023

1 Bài toán 1.

Cho mảng A đã được sắp xếp có N phần tử. Tìm một phần tử x trong A.

- Input: A, x.
- Output: Vị trí nằm trong $A = x$, nếu không tìm thấy return none.

Test dữ liệu:

- $N = 10, 20, 30, ...$
- Tạo ngẫu nhiên A đã được sắp xếp và phần tử x.

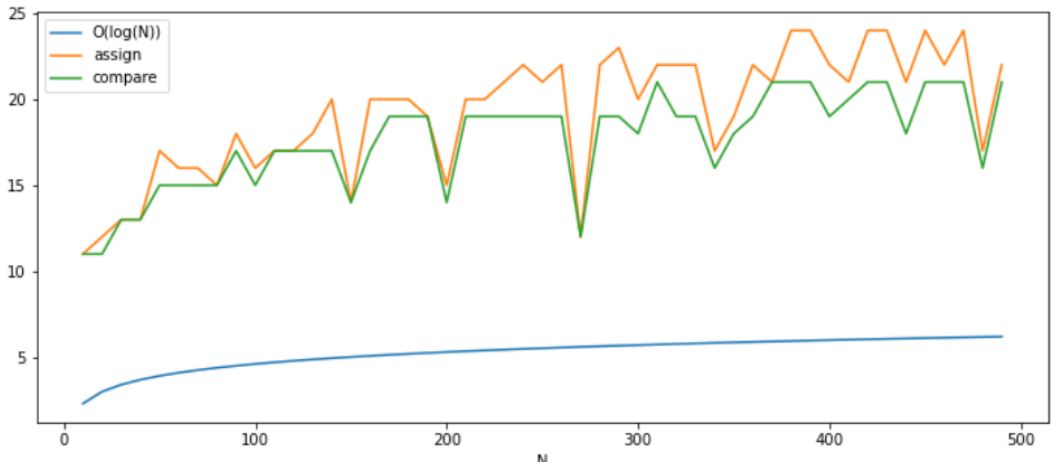
Bài 1
Trong bài toán này, ta dùng thuật toán Binary Search để tìm vị trí của x trong A, nếu không tìm thấy thì trả về -1

Algorithm 1 Binary Search

```
1: function BINARY-SEARCH(arr, x)
2:     low ← 0; high ← (length(arr) − 1); mid ← 0;
3:     while low ≤ high do
4:         mid ← ⌊(high + low)/2⌋;
5:         if arr[mid] < x then
6:             low ← mid + 1;
7:         else if arr[mid] > x then
8:             high ← mid − 1;
9:         else
10:            return mid;
11:        end if
12:    end while
13:    return −1;
14: end function
```

Sử dụng thuật toán trên cho bài toán này, với mảng A và x được tạo ngẫu nhiên, ta có kết quả thực thi như sau:
Hơn nữa, thuật toán này có độ phức tạp là $O(N \log_(N))$ ta kiểm chứng bằng cách vẽ đồ thị biểu diễn số phép gán và số phép so sánh của thuật toán khi N thay đổi như sau:

```
              Exercise 1
****************************************
>> Array A:
 [ 18  99 111 121 165 191 244 315 373 517 531 606 615 704 718 826 845 866
 918 980]
365 is not in A.
```



Ta có thể thấy rằng 2 đường biểu diễn số phép gán và số phép so sánh của thuật toán Binary Search có dạng tương đối giống với đường $O(N \log_N)$. Như vậy, độ phức tạp của thuật toán Binary Search là $O(N \log_N)$.

Bài 2 Cho tập hợp S có N phần tử khác nhau, $S(i) \in \{1, 2, ..., 1000\}$. Viết chương trình tìm phần tử nhỏ nhất thứ k trong tập S $(1 \leq k \leq N)$
Dữ liệu test:

- $N = 10, 20, 30, ..., 100$ và $k = 5$.
- Với N cố định, tạo ngẫu nhiên tập S, trong đó $S(i) \in \{1, 2, ..., 1000\}$ và tính thời gian trung bình để $S(i) \neq S(j), \forall i \neq j$

lời giải

Ý tưởng cho bài toán trên là ta đi sắp xếp lại mảng S theo thứ tự tăng dần, sau đó ta lấy ra phần tử thứ$k - 1$ trong mảng đã sắp xếp. Thuật toán sắp xếp ở đây, ta sẽ dùng thuật toán Quick Sort.
Cài đặt thuật toán Quick Sort
Khi đó ta có hàm tìm phần tử nhỏ nhất thứ k trong tập S như sau
Kết quả thực thi thuật toán trên như sau
Trong khi đó, tập S được tạo ngẫu nhiên như sau
Ta nhận thấy rằng, thuật toán trên với trường hợp tối ưu nhất thì vòng while chạy đúng 1 lần nếu giá trị mới được tạo ngẫu nhiên không trùng với các phần tử từ vị trí $i - 1$ trở về trước của tập S, khi đó thuật toán này có độ phức tạp $O(N)$. Với trường hợp tệ nhất là mỗi lần tạo ngẫu nhiên đều trùng với từng phần tử từ vị trí $i - 1$ trở về trước của tập S, tức là phải tốn $i - 1$ lần mới tạo ra được giá trị ngẫu nhiên mới, khi đó thuật toán có độ phức tạp là $O(N \log_N)$. Ta có minh hoạ để kiểm chứng điều trên như sau
Ta thấy rằng các đường biểu thị số phép gán và số phép so sánh của thuật toán tạo tập S này có dạng tương đối giống với đường $O(N)$.

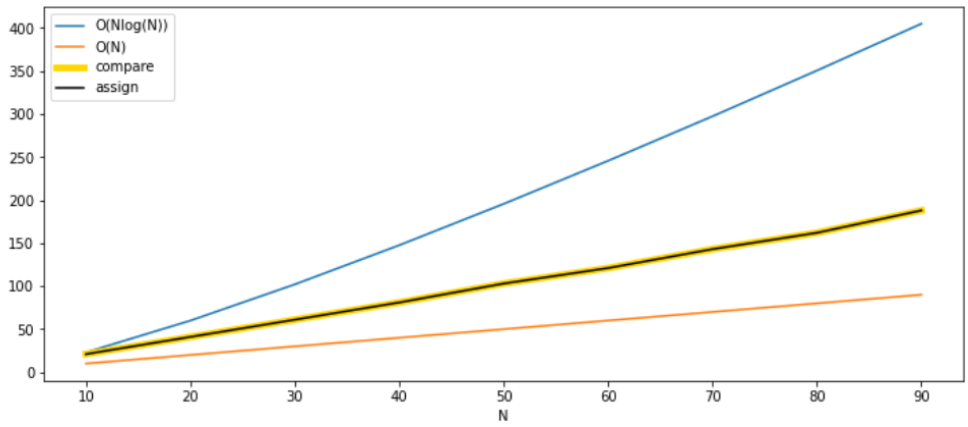
Entrée [4]:
```python
def quickSort(a_list):
    def _quicksort(a_list, low, high):
        # must run partition on sections with 2 elements or more
        if low < high:
            p = partition(a_list, low, high)
            _quicksort(a_list, low, p)
            _quicksort(a_list, p+1, high)
    def partition(a_list, low, high):
        pivot = a_list[low]
        while True:
            while a_list[low] < pivot:
                low += 1
            while a_list[high] > pivot:
                high -= 1
            if low >= high:
                return high
            a_list[low], a_list[high] = a_list[high], a_list[low]
            low += 1
            high -= 1
    _quicksort(a_list, 0, len(a_list)-1)
    return a_list
```

```python
def find_kth_smallest(arr, k):
    sorted_arr = quickSort(arr)
    return sorted_arr[k-1]
```

```
              Exercise 2
**************************************
>> Array A:
 [126, 796, 803, 865, 458, 50, 182, 151, 978, 740, 233, 757, 904, 467, 75, 97, 388, 315, 862, 36]
>> Sort A:
 [36, 50, 75, 97, 126, 151, 182, 233, 315, 388, 458, 467, 740, 757, 796, 803, 862, 865, 904, 978]
>> The 5th smallest in A: 126
```

Entrée [3]:
```python
def random_set(low, high, size):
    compare = assign = 0
    array = []
    for i in range(size):
        temp = np.random.randint(low,high)
        while temp in array[:i]:
            temp = np.random.randint(low,high)
            compare += 1
            assign += 1
        array.append(temp)
        assign += 2
        compare += 2

    compare += 1
    assign += 1
    return array, compare, assign
```


O(Nlog(N))
O(N)
compare
assign
400
350
300
250
200
150
100
50
0
10
20
30
40
50
60
70
80
90
N