# Binary Search

locked

Problem | Submissions | Leaderboard | Discussions

Binary search merupakan pencarian yang cepat, karena dalam setiap iterasinya, interval dari pencarian akan dibagi dua. Algoritma ini memiliki kompleksitas waktu O(log n), lebih cepat dibandingkan algoritma linear search (O(n))

Untuk meningkatkan pemahaman anda mengenai bagaimana cara kerja algoritma dengan kompleksitas waktu O(log n), implementasikan algoritma Binary Search terhadap array of integer dan hitunglah jumlah operasi yang dilakukan!

Contoh penghitungan jumlah operasi algoritma Linear Search menggunakan C++:

```
int main() {
    // Jumlah operasi untuk input dan output tidak perlu dihitung
    // n: panjang array
    // x: elemen yang dicari
        int n, x;
    cin >> n >> x;

    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Mulai hitung jumlah operasi
    int operasi = 1, i = 0;
    while (arr[i] != x && i < n) {
        operasi++;
        i++;
    }

    // Outputkan -1 dan tambahkan jumlah operasi apabila elemen tidak ditemukan
    if (i >= n) {
        operasi++;
        i = -1;
    }

    cout << i << ' ' << operasi;
    return 0;
}
```

### Input Format

Baris pertama berisikan dua buah integer yang merupakan panjang array (n) dan elemen yang dicari (x). x belum tentu ada di dalam array Baris kedua berisikan n buah integer yang merupakan elemen-elemen dari array (arr[i]). Karena Binary Search mengharuskan array terurut, maka input array **dipastikan terurut dari bilangan terkecil hingga terbesar**

Solved: 175
Attempted: 194

### Constraints

$1 \le n, x, arr[i] \le 1000$

### Output Format

Dua buah integer, dimana integer pertama merepresentasikan lokasi elemen tersebut dalam array (0 indexed, **outputkan -1** apabila elemen tidak ditemukan) dan jumlah operasi yang dilakukan dalam Binary Search tersebut. Apabila elemen tidak ditemukan, tambahkan 1 ke jumlah operasi (bisa melihat sample case 1)

### Sample Input 0

```
5 8
1 2 4 8 16
```

### Sample Output 0

```
3 2
```

### Explanation 0

Ketika mencari nilai 8 menggunakan binary search, operasi yang dilakukan adalah sebagai berikut:

- operasi = 0
- 1 2 **4** 8 16 → midIdx = 2, arr[midIdx] = 4, operasi = 1
- *1 2 4* **8** 16 → midIdx = 3, arr[midIdx] = 8 = x, operasi = 2

x (8) ditemukan di index ketiga pada operasi kedua

Output: 3 2

### Sample Input 1

```
3 6
1 2 3
```

### Sample Output 1

```
-1 3
```

### Explanation 1

Ketika mencari nilai 6 menggunakan binary search, operasi yang dilakukan adalah sebagai berikut:

- operasi = 0
- 1 **2** 3 → midIdx = 1, arr[midIdx] = 2, operasi = 1
- *1 2* **3** → midIdx = 2, arr[midIdx] = 3, operasi = 2
- l > r (interval habis), x tidak ditemukan, operasi = 3

x (6) tidak ditemukan, outputkan -1

Output: -1 3

Python 3

```python
def binary_search(arr, target):
    left = 0
    right = len(arr) - 1
    operations = 0

    while left <= right:
        mid = (left + right) // 2
        operations += 1

        if arr[mid] == target:
            return mid, operations

        elif arr[mid] < target:
            left = mid + 1


        else:
            right = mid - 1


    return -1, operations + 1

def pemahaman_binary():
    input_panjang_dicari = input()
    panjang, dicari = map(int, input_panjang_dicari.split())
    input_arr = input()
    arr = list(map(int, input_arr.split()))
    arr.sort()
    binary_search(arr, dicari)
    print(str(binary_search(arr, dicari)[0]) + " "+str(binary_search(arr, dicari)[1]))

pemahaman_binary()
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code