Search and Sort

Luke Wu

Search

在一個任意排序的 Array 內找尋特定數字

ex: 下列 Array 內是否有 38

[55, 10, 35, 1, 2, 25, 37]

在一個 Sorted Array 內找尋特定數字

ex: 下列 Array 內是否有 38

[1, 2, 10, 25, 35, 37, 55]

Which one is better?

Time Complexity 時間複雜度

- Machine Independent - 運算次數
- 評估**運算次數**與**輸入**之間的關係

Asymptotic Notation - Big “Oh”

- f(n)=O(g(n)) iff
  - ∃ positive const. c and n_0, ∍ f(n) ≤ cg(n) ∀ n, n ≥ n_0
  - e.g.
    - **3n+2 =O(n)** **3n+2 ≤ 4n for all n ≥ 2**
    - **$10n^2+4n+2=O(n^2)$** **$10n^2+4n+2 \leq 11n^2$, for all n ≥ 10**
    - **$3n+2 = O(n^2)$** **$3n+2 \leq n^2$ for all n ≥ 4**

* g(n) should be a ***least upper bound***

Time Complexity

- Linear Search - O(n)
- Binary Search(Under sorted array) - O(log n)

* In Complexity, **log** ususlly stand for **log to the base 2**

```
if (n > 10) {
      print(“n is bigger than 10”)
}
```

O(n)

```
for (i = 0; i < n; i++) {
       print(“Hello world”)
}
```

```
for (i = 0; i < 1000; i++) {
        print("Hello world")
}
```

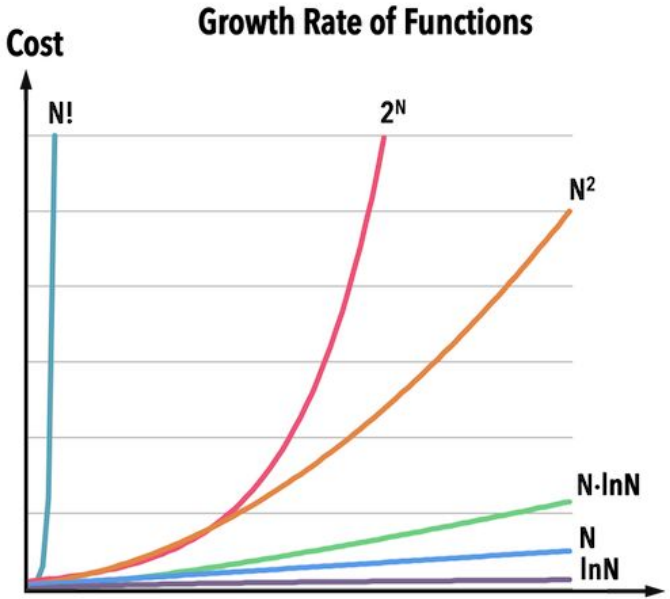
O(n)

```
for (i = 0; i < n; i++) {
        print("AppWorks School")
}
```

```
for (i = 0; i < n; i++) {
    for (j = 0; j < i; i++) {
        print("AppWorks School")
    }
}
```

O(n^2)

Complexity Compare



Sort

Selection Sort

Pseudo Code

提供足夠詳細的程式執行流程以描述程式的每一個執行步驟，藉以提供程式設計師在接下來把程式以特定的程式語言完成

Insertion Sort

Merge Sort

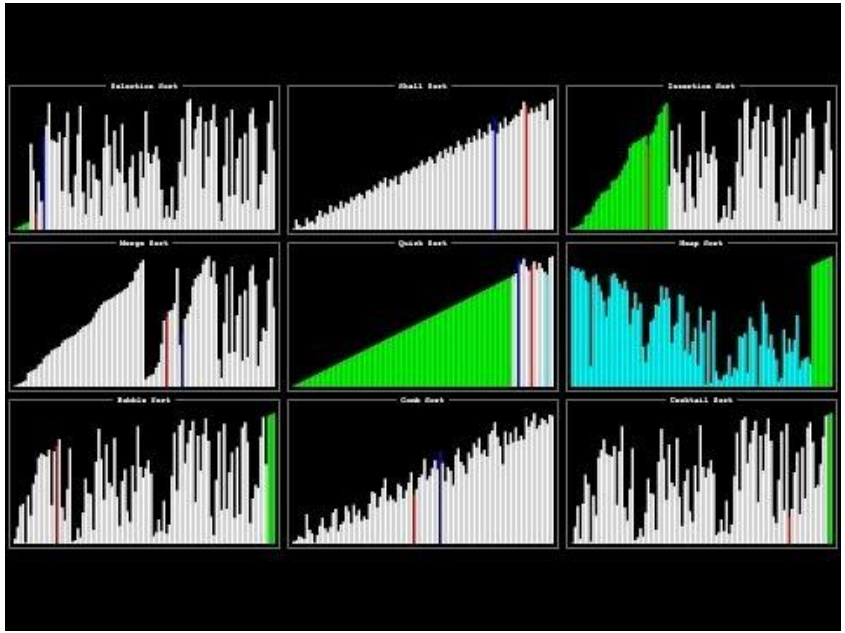
Time Complexity

- Selection Sort - O(n^2)
- Insertion Sort - O(n^2)
- Merge Sort - O(n * log n)

Time Complexity

- Best Case
- Worst Case
- Average Case

Time Complexity



Question - Why Do We Take The Fisrt Senario?

- Merge Sort + Binary Search = O(n * log n) + O(log n)
- Linear Search = O(n)

Reference

- http://alrightchiu.github.io/SecondRound/mu-lu-yan-suan-fa-yu-zi-liao-jie-gou.html
- NCTU Data Structure - 彭文志 教授
- NTHU Data Structure - 韓永楷 教授