CSI213 - Lab 4: Algorithm Analysis

Huang Kaisheng <2020215138@stu.cqupt.edu.cn>

November 4, 2021

Contents

1 A binary search of n elements.

Answer: $\mathrm{O}(\log n)$

A prerequisite of binary search is that the sequence should be sorted. We assume that the sequence is ascending ordered.

The algorithm always selects the middle element and divide the sequence into two parts. If the middle element is the target it returns, else it recursively runs itself on the left part or right part, chosen by comparing the target and the middle element. If less, it chooses the left part. If greater, it chooses the right part.

Pseudo Code:

Algorithm 1 Binary search

Require: $x \in a_n$
Ensure: a_n is ascending ordered.

```
1: procedure Find(x: integer, l := 0, r := n: non-negative integers)
2:     i := ⌊r/2⌋
3:     if a_i = x then return i
4:     else if a_i < x then return Find(x, 0, i)
5:     else return Find(x, i + 1, n)
6:     end if
7: end procedure
```

2 A linear search to find the smallest number in a list of n numbers.

Answer: $\mathrm{O}(n)$

This algorithm visits every position to find the smallest one.

Pseudo Code:

Algorithm 2 Find smallest number

```
1: i := 1, m := a_0                    ▷ m will be the smallest number
2: while i < n do
3:     m := min(m, a_i)
4:     i := i + 1
5: end while
```

3 An algorithm that prints all bit strings of length n.

Answer: $\mathrm{O}(2^n)$

Each bit of an n-length bit string have two possible values: 0, 1.

Pseudo Code:

Algorithm 3 Generate bit strings

```
1: procedure GenBitString(n, i := 0, bitString := 0: non-negative integers)
2:     if i = n then
3:         Print(bitString)
4:     else
5:         GenBitString(n, i + 1, (bitString ≪ 1)|1)
6:         GenBitString(n, i + 1, bitString ≪ 1))
7:     end if
8: end procedure
```

4 The number of print statements (Question #4)

Answer: $\mathrm{O}(n)$

For each iteration of i, the inner loop variable j increases from the last j to i. On every iteration, j is always equal to i. So it actually runs n times.

5 The number of print statements (Question #5)

Answer: $\mathrm{O}(\log n)$

The loop variable n be a half of n each time of iteration. So it can only run $\lfloor \log n \rfloor$ times.

6 An iterative algorithm to compute n!, (counting the number of multiplications).

Answer: $\mathrm{O}(n)$

It iterates from 1 to n, n times iterated.

Pseudo Code:

Algorithm 4 Compute $n!$

1: $i := 1, m := 1$ ▷ m will be the $n!$
2: **while** $i \leq n$ **do**
3: $\quad m := m \times i$
4: $\quad i := i + 1$
5: **end while**

7 An algorithm that finds the average of n numbers by adding them and dividing by n.

Answer: $\mathrm{O}(n)$

It just make a sum of the sequence and make a division.

Pseudo Code:

Algorithm 5 Calculate average

1: $i := 0, sum := 0$
2: **while** $i < n$ **do**
3: $\quad sum := sum + a_i$
4: $\quad i := i + 1$
5: **end while**
6: $avg := \frac{sum}{n}$

8 An algorithm that prints all subsets of size three of the set 1, 2, 3, ..., n.

Answer: $\mathrm{O}(n^3)$

The question can be converted to the solutions to choose three elements in n elements. So the amount should be

$$\mathrm{C}_n^3 = \frac{n!}{3!(n-3)!} = \frac{n(n-1)(n-2)}{3!}$$

Pseudo Code:

Algorithm 6 Print three-size subsets

```
1: i := 0, j := i, k := j
2: while i < n do
3:     j := i + 1
4:     while j < n do
5:         k := j + 1
6:         while k < n do
7:             Print(a_i, a_j, a_k)
8:             k := k + 1
9:         end while
10:        j := j + 1
11:    end while
12:    i := i + 1
13: end while
```

9 The best-case analysis of a linear search of a list of size n (counting the number of comparisons).

Answer: $\mathrm{O}(1)$

For example, there is a list:

$$a_n = 1, 2, 3, 4, 5, 6, 7$$

And the wanted one is the first element 1, so only 1 comparison is needed.

10 The worst-case analysis of a linear search of a list of size n (counting the number of comparisons).

Answer: $\mathrm{O}(n)$

For example, there is a list:

$$a_n = 1, 2, 3, 4, 5, 6, 7$$

And the wanted one is the last element 7, so we have to compare for n times.

11 Give a big-O estimate for the number of operations (Question #11)

Answer: $\mathrm{O}(n^2)$

Inside the loop, $t + 2it$ will result in 3 operations (two multiplications and one addition) each iteration.

About the loop, I have asked the TA. She says that I can assume that i increases by 1 from n to n^2. So it takes $n^2 - n$ operations.

Therefore, this algorithm segment results in $(n^2 - n) \times 3$ operations totally.

12 Give a big-O estimate for the number of operations (Question #12)

Answer: $\mathrm{O}(n^2)$

Inside the inner loop, $(it+jt+1)^2$ will result in 5 operations (two multiplication between i, j and t, two addition and a multiplication for square) each iteration.

The inner loop will execute n times, and the outer loop will also execute n times.

In total, $5 \times n \times n$ operations will be result in.