Problem Set II

Huy Quang Lai
132000359

Texas A&M University

17 September 2022

Introduction to Analysis of Algorithms

Problem 1

$$\frac{2}{N}, 12, \sqrt{N}, N, N\log\log N, N\log N, N\log N^2,$$
$$N\log^2 N, N^{1.5}, N^2, N^2\log N, N^3, 2^{\frac{N}{2}}, 2^N$$

$N^{1.5}$, N^2, and N^3 grow polynomially.
$2^{\frac{N}{2}}$ and 2^N grow exponentially.
$N\log N$ and $N\log N^2$ grow linearithmically

Problem 2

A. $O\left(2^{2^N}\right)$

B. $O\left(\log_2(\log_2 D)\right)$

C. $O\left(\log_2\log_2 D\right)$

Problem 3

$3\ min = 1.8 \times 10^8\ \mu s$

A. linear $O(N)$
$C(100) = 700 \rightarrow C = 7$
2.7×10^7 items

B. linearithmic $O(N \log N)$
$C(100)\log_2(100) = 700 \rightarrow C = \dfrac{7}{2\log_2 10}$
$\dfrac{7}{2\log_2 10} x \log_2 x = 1.8 \times 10^8$
Using Wolfram-Alpha to solve:
7.48×10^6 items

C. quadratic $O(N^2)$
$C(100^2) = 700 \rightarrow C = 0.07$
= 5.07×10^4 items

D. cubic $O(N^3)$
$C(100^3) = 700 \rightarrow C = 0.0007$
6.40×10^3 items

E. exponential $O(2^N)$
$C(2^{100}) = 1.8 \times 10^8 \rightarrow C = 5.52 \times 10^{-28}$
1.18×10^1 items

Problem 4

A. `Bar` is more guaranteed to run faster for values of $N < 100$

B. `Foo` is more guarenteed to run faster for values of $N > 100000$

C. $221N\log_2 N = 3N^2 \Rightarrow N = 1$

D. `Bar` can run faster than `Foo` if the input size is sufficiently small enough

Run Time Calculations

Problem 1

1. Fragment 1
   $O(N)$

```
void frag1(size_t n) {
    size_t sum = 0;
    for (size_t i = 0; i < n; ++i)
        sum = sum + 1;
}
```

Figure 1: Fragment 1

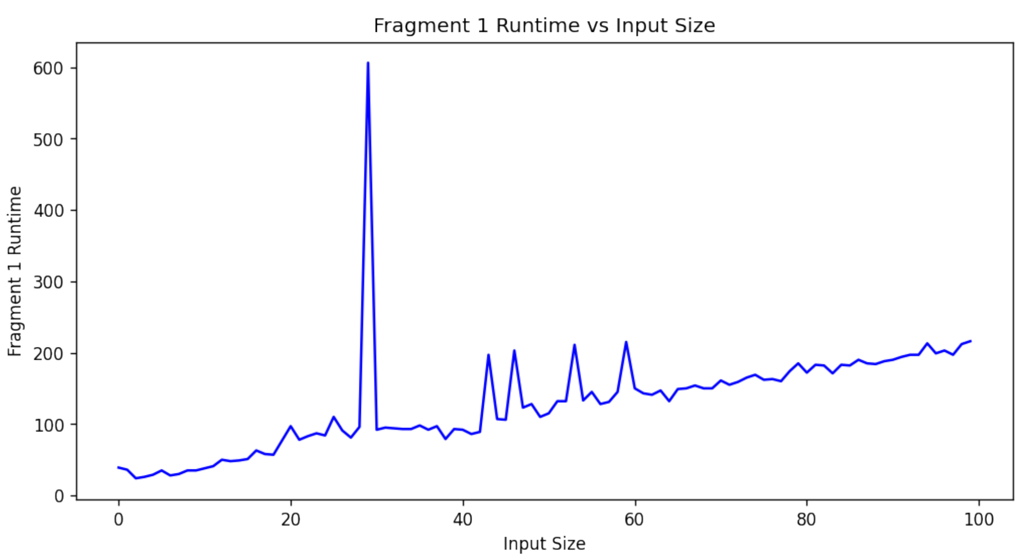

Figure 2: Fragment 1 Runtime

2. Fragment 2
   $O(N^2)$

```
void frag2(size_t n) {
    size_t sum = 0;
    for (size_t i = 0; i < n; ++i)
        for (size_t j = 0; j < i; ++j)
            sum = sum + 1;
}
```

Figure 3: Fragment 2

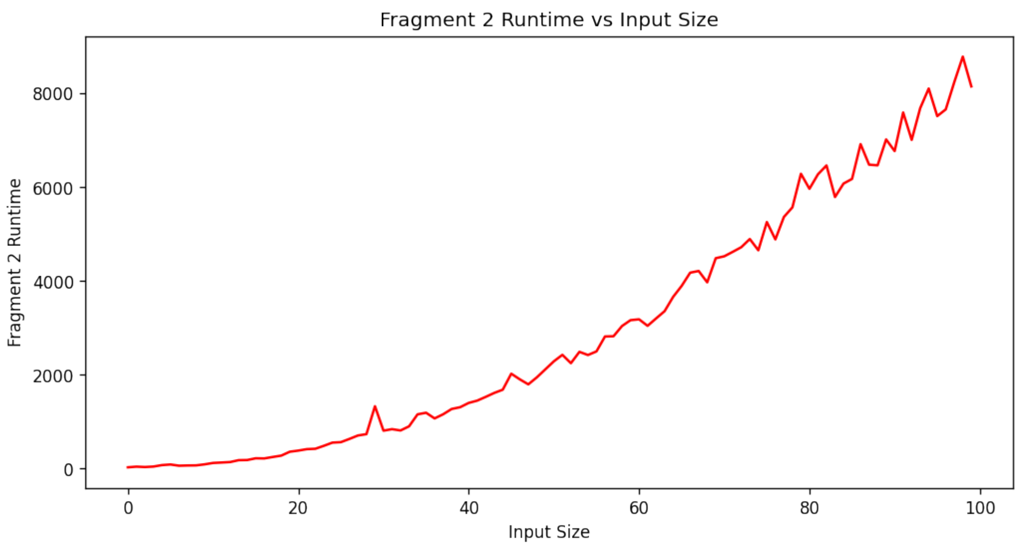

Figure 4: Fragment 2 Runtime

3. Fragment 3
   $O(N^5)$

```
void frag3(size_t n) {
    size_t sum = 0;
    for (size_t i = 0; i < n; ++i)
        for (size_t j = 0; j < i * i; ++j)
            for (size_t k = 0; k < j; ++k)
                sum = sum + 1;
}
```

Figure 5: Fragment 3

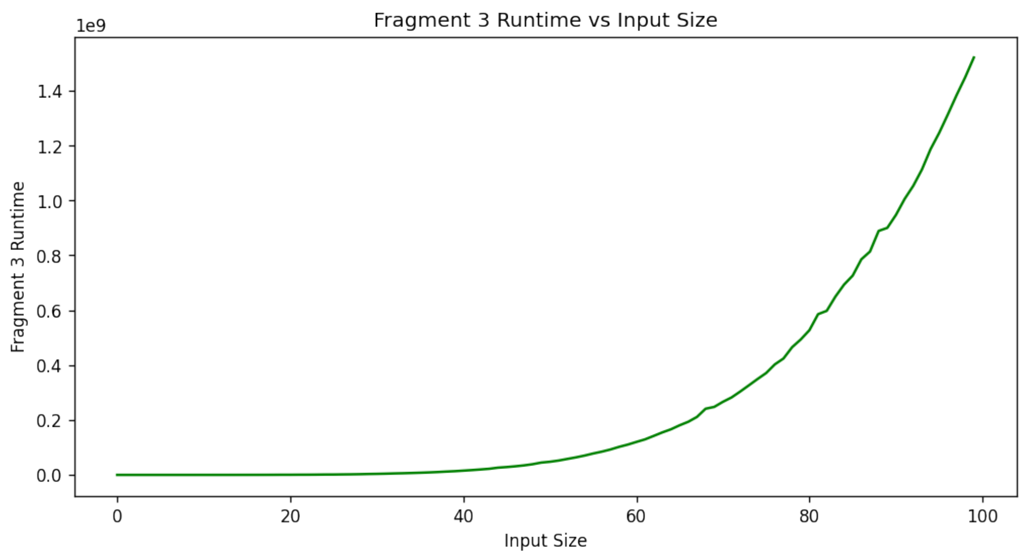

Figure 6: Fragment 3 Runtime

Problem 2

For each of the following operations, write out (e.g. in pseudocode) the algorithm you typically use for hand-calculations and give a big-O estimate of the time complexity of that algorithm.

A. Add two N-digit integers

```
def add(int a, int b):
    Let x = min(a, b)
    Let y = max(a, b)
    Let carry = 0
    Let digits = []

    while x > 0:
        Let dx = x % 10
        Let dy = y % 10
        Let sum = dx + dy + carry
        if sum > 10:
            sum %= 10
            carry = 1
        else:
            carry = 0
        end

        digits.append(sum)
        x //= 10
        y //= 10
    end

    print digits in reverse
end
```

$O(n)$

B. Multiply two N-digit integers

```
def mult(a, b):
    Let product = 0;
    while (b > 0):
        product = add(product, a)
        b = b - 1
    end
    return product
end
```

$O(N^2)$

C. Divide two N-digit integers

```
def divide(a, b):
    Let quotient = 0;
    Let neg_b = mult(-1, b)
    if (b < a):
        return 0
    end
    while (a > 0):
        quotient = quotient + 1
        a = add(a, neg_b)
    end
    return quotient
end
```

$O(N^2)$

Problem 3

Use big-O to estimate how much time is required to compute the following function:

$$f(x) = \sum_{i=0}^{N} a_i x^i$$

A. $O(N)$

B. $O(\log N)$

C. $O(N)$

Puzzle Problem [optional]

Show that X^{62} can be computed with only 8 multiplications
$x^2 \to x^4 \to x^6 \to x^{12} \to x^{24} \to x^{30} \to x^{31} \to x^{62}$

Extra Challenge

What is the smallest possible power k such that X^k required at least eight multiplications.