Problem Set VII

Huy Quang Lai
132000359

Texas A&M University

6 November 2022

Problem 1

A. Let L be a list of size n
The total number of permutations is $n!$, giving the decision tree a total of $n!$ leaves. The decision tree with I leaves will have a max depth of $\lceil \log_2 I \rceil$. With this the decision tree for the sorting algorithm will be $\lceil \log_2 n! \rceil$

$$\begin{aligned}
\log_2 n! &= \log_2(n(n-1)(n-2)\cdots(2)(1)) \\
&= \log_2 n + \log_2(n-1) + \log_2(n-2) + \cdots + \log_2(2) + \log_2(1) \\
&\geq \log_2 n + \log_2(n-1) + \log_2(n-2) + \cdots + \log_2\left(\frac{n}{2}\right) \\
&\geq \frac{n}{2}\log_2\left(\frac{n}{2}\right) \\
&\geq \frac{n}{2}\log_2 n - \frac{n}{2} \\
&= \Omega(n \log_2 n)
\end{aligned}$$

Therefore, a list of size 4 will require 5 comparisons.

B. The algorithm for list $L = \{A, B, C, D\}$

```
void sort(int[4]& values) {
    if (values[0] > values[1])
        swap(values[0], values[1]);
    if (values[2] > values[3])
        swap(values[2], values[3]);
    if (values[0] > values[2])
        swap(values[0], values[2]);
    if (values[1] > values[3])
        swap(values[1], values[3]);
    if (values[1] > values[2])
        swap(values[1], values[2]
}
```

C. $L = \{4, 2, 3, 1\}$

1. $L = \{2, 4, 3, 1\}$
2. $L = \{2, 4, 1, 3\}$
3. $L = \{1, 4, 2, 3\}$
4. $L = \{1, 3, 2, 4\}$
5. $L = \{1, 2, 3, 4\}$

Problem 2

```
void sort(std::vector<Key>& values) {
    size_t f{}, m{}, t{};
    for (const Key& key : values) {
    switch (Key) {
        case true:
            ++t;
            break;
        case false:
            ++f;
            break;
        case maybe:
            ++m;
            break;
        default:
            continue;
        }
    }

    values.clear();
    values.insert(values.end(), f, false);
    values.insert(values.end(), m, maybe);
    values.insert(values.end(), t, true);
}
```

Problem 3

$O(N^2)$

```
bool findPair(vector<int>& values, const int& k) {
    for (auto i = values.cbegin(); i != values.cend(); ++i)
        for (auto j = i; j != values.cend(); ++j)
            if (*i + *j == k)
                return true;

    return false;
}
```

$O(N \log N)$

```
bool findPair(vector<int>& values, const int& k) {
    sort(values.begin(), values.end());
    for (auto i = values.cbegin(); i != values.cend(); ++i)
        if (binary_search(values.cbegin(), values.cend(), k - *i))
            return true;

    return false;
}
```

Runtime Graphs on the next page

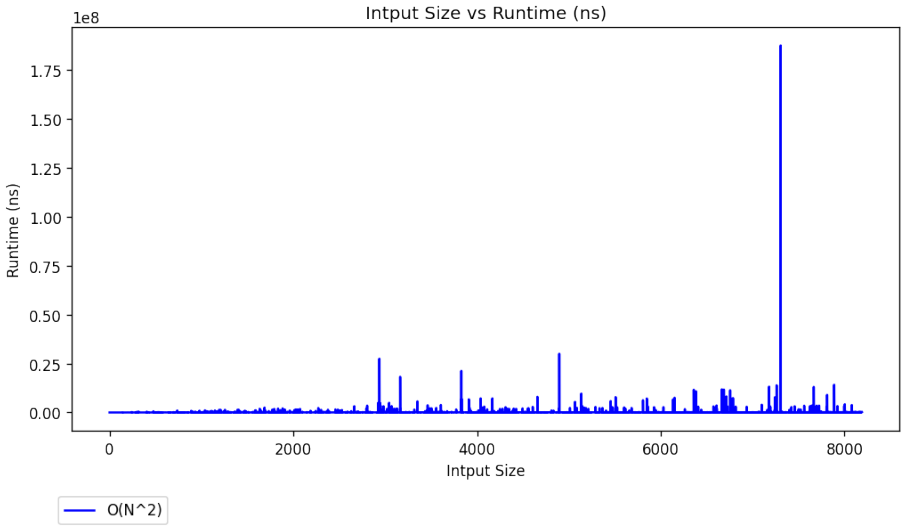

Figure 1: $O(N^2)$

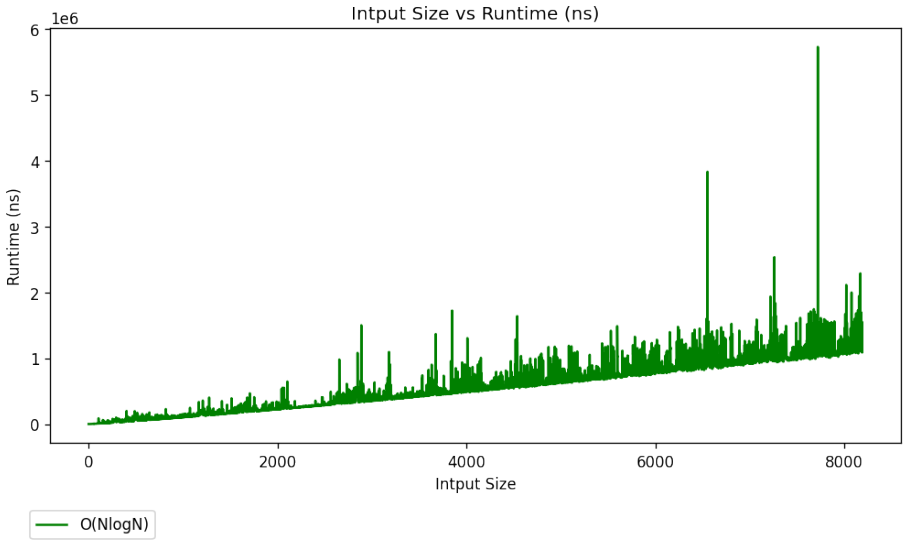

Figure 2: $O(N \log N)$

Problem 4

A. Use a binary search to find N.

B. Start at index $\sqrt{N}$. Until your guess is too high or your index is too large, increment by $\sqrt{N}$. Once the index become too large, return to the previous largest guess and linear search the remaining list. Through this algorithm, the number of guesses is $O(\sqrt{N})$

C. Using the algorithm in part B. If the number you are searching for is a multiple of $\sqrt{N}$, then no linear searching is required. This makes the algorithm $\Omega(\sqrt{N})$ in this case.

D. Binary search the list using all the allowed high guess. Once all guesses are used, linear search the remaining portion of the list. The time complexity of this algorithm would be $O\left(\frac{N}{2^g}\right)$. In the worse case, all guesses will be too high using all guesses which will divide the list in half g times. Linear searching the remaining part of the list, from the beginning to $\frac{N}{2^g}$.