CSC111 Lecture 22: Average-Case Running Time Analysis

Hisbaan Noorani

March 31, 2021

Contents

Our worksheet exercises today will focus on analysing the average-case running time for the following implementation of the *linear search* algorithm:

```python
def search(lst: list, x: Any) -> bool:
    """Return whether x is in lst."""
    for item in lst:
        if item == x:
            return True
    return False
```

1 Exercise 1: Counting binary lists

In lecture, we began an average-case running time analysis for `search` on binary lists, i.e., lists where each element is either 0 or 1. Let $n \in \mathbb{N}$. We define the set of inputs $\mathcal{I}_n$ to be the set of binary lists of length n.

1. Suppose $n = 3$. Write down $\mathcal{I}_3$, i.e., the set of binary lists of length 3. You should have **eight** lists in total.

   `{[0, 0, 0,], [0, 0, 1], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0], [1, 1, 1]}`

2. Now suppose n is an arbitrary natural number. Find an expression (in terms of n) for $|\mathcal{I}_n|$, the size of $\mathcal{I}_n$. (You don't need to formally prove that this expression is correct, but try to briefly justify this expression for yourself.)

   2^n. Here we have $n = 3$, and we were able to generate 8 lists. We cannot have done more than this.

   $$\underbrace{0/1 \quad 0/1 \quad 0/1 \quad \dots \quad 0/1}_{n}$$

2 Exercise 2: Partitioning $\mathcal{I}_n$

Here is the definition of the partitions of $\mathcal{I}_n$ from lecture.

- For each $n \in \mathbb{N}$ and each $i \in \{0, 1, \ldots, n-1\}$, let $S_{n,i}$ denote the set of all binary lists of length n where the first `0` occurs in index i. More precisely, every list `lst` in $S_{n,i}$ satisfies the following two properties:

  1. `lst[i] = 0=`
  2. For all $j \in \{0, 1, \ldots, i-1\}$, `lst[j] = 1=`

- For each $n \in \mathbb{N}$, let $S_{n,n}$ denote the set of binary lists of length n that do not contain a `0` at all.

1. To make sure you understand the definitions of these partitions, write down the corresponding set of binary lists for each of the following.

   (a) $S_{3,0}$
   `{[0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1]}`

   (b) $S_{3,1}$
   `{[1, 0, 0], [1, 0, 1]}`

   (c) $S_{3,2}$
   `{[1, 1, 0]}`

   (d) $S_{3,3}$
   `{[1, 1, 1]}`

2. Now find expressions for each of the following. Once again, make sure you can briefly justify your answers, but no need for formal proofs.

   (a) $|S_{n,n}|$
   $\underbrace{1 \quad 1 \quad 1 \quad \ldots \quad 1}_{n} \implies 1$

   (b) $|S_{n,i}|$, where $i \in \{0, 1, \ldots, n-1\}$ (your expression should be in terms of i and n)
   $\underbrace{1 \quad 1 \quad 1 \quad \ldots \quad 1 \quad 0}_{i+1} \quad \ldots \quad \underbrace{0/1 \quad \ldots \quad 0/1}_{n-i-1} \implies 2^{-i-1}$

3. Recall that the purpose of this partitioning is that all input lists `lst` in the same partition have the same running time for `search(lst, 0)`:

   - Every input in $S_{n,i}$ (for $i \in \{0, 1, \ldots, n-1\}$) takes $i+1$ steps
   - Every in $S_{n,n}$ takes $n+1$ steps (technically this is compatible with the previous bullet point, setting $i = n$)

Using this and your answers to the previous question, simplify the expression below. *Tips:* Use the formula $\sum_{i=0}^{m-1} i \cdot r^i = \frac{m \cdot r^m}{r-1} - \frac{r(r^m - 1)}{(r-1)^2}$, valid for all $m \in \mathbb{Z}^+$ and $r \in \mathbb{R}$ where $r \neq 1$.
b. Use the change of variable $i' = i + 1$ to help see how to use the above formula.
Expression to simplify:

$$\sum_{i=0}^{n} |S_{n,i}| \times \big(\text{running time of } \texttt{search(lst, 0)} \text{ when } \texttt{lst} \in S_{n,i}\big)$$

Let $m = n + 1$, $r = \frac{1}{2}$

$$\begin{aligned}
&= \sum_{i=0}^{n} |S_{n,i}| \times (i+1) \\
&= \sum_{i=0}^{n-1} \left[2^{n-i-1} \times (i+1)\right] + 1 \times (n+1) \\
&= \sum_{i'=1}^{n} \left[2^{n-i'} \times i'\right] + (n+1) \\
&= 2^n \left(\sum_{i'=1}^{n} \left(\frac{1}{2}\right)^{i'} \times i'\right) + (n+1) \\
&= 2^n \left[\sum_{i'=1}^{n} \frac{(n+1)\left(\frac{1}{2}\right)^{n+1}}{\frac{1}{2} - 1} - \frac{\frac{1}{2}\left(\left(\frac{1}{2}\right)^n + 1\right) - 1}{\left(\frac{1}{2} - 1\right)^2}\right] + (n+1) \\
&= \dots \\
&= 2^{n+1} - 1
\end{aligned}$$

This last expression gives you the total running time of `search` over all inputs in $\mathcal{I}_n$! We'll continue lecture by using this to calculate the average-case running time of `search` for this input set, but if you still have time feel free to work it out yourself.

3 Additional exercises

1. Generalize our average-case running time analysis for `search` for the input set consisting of all lists of length n where every element is a 0, 1, or 2, and `x = 0`.

2. Let $m \in \mathbb{Z}^+$. Generalize our average-case running time analysis for `search` for the input set consisting of all lists of length n where every element is a number between 0 and $m - 1$, inclusive, and `x = 0`.

3. Analyse the average-case running time of `search` when the input set is the list of all **permutations** of the numbers $\{0, 1, \dots, n-1\}$ and `x = 0`.

4. Consider the following algorithm for checking whether a string is a palindrome:

```python
def is_palindrome(s: str) -> bool:
    """Return whether s is a palindrome."""
    mid = len(s) // 2
    for i in range(0, mid):
        if s[i] != s[len(s) - 1 - i]:
            return False

    return True
```

Analyse the average-case running time of `is_palindrome` on the input set $\mathcal{I}_n$ consisting of all *binary strings* of length n (i.e., strings that contain only the characters `'0'` and `'1'`).