Algorithm Analysis: Induction

CSCI241

October 20, 2022

1 Algorithm

1.1 Problem → Algorithm → Program

- **Algorithm:** A finite set of instructions that if followed, a particular task can be accomplished.
- **Computer Algorithm:** A step by step procedure used by a computer to solve a problem
  - has input and output
  - each step is unambiguous
  - terminates in a finite number of steps in a resonably short period of time

1.2 Criteria for algorithm analysis

- Correctness
- Time/Computational Complexity
- Space Complexity
- Simplicity & Readability
- Optimality

1.3 Time Complexity: Big O notation

See section 6.2

2 Example 1: Sequential Search

Note that we use recursion here.

2.1 Algorithm description of sequential search

- **Problem**: Is item **x** in array **s** of **n** items?
- **Inputs: s, k, n, s** where
  - s: array of n items to process
  - k: index of the kth element in s
  - n: number of items in s
  - x: target to search for

- **outputs:** the least k with s[k]=x or -1
- **Algorithm in Pseudocode**

```
Search(s,k,n,x)
    if(k>n)
        return -1
    if(s[k]=x)
        return k
    else
        Search(s,k+1,n,x)
EndSearch
```

2.2 Algorithm Analysis of sequential search

- **Correctness**: In general, tedious, difficult, and complicated. For recursion algorithms, we need to ask 3 questions:
  - Q1: Is there base(s)? (Does the recursion end?)
  - Q2: Does the recursion call solve a smaller part of the or original problem?
  - Q3: Does the whole procedure work assuming the recursion works?
- **Time complexity**:
  - difficult to measure
  - focus on the key operation(s) in term of the input size n
  - in search algorithm, the key operation is comparison.
  - **best case:**
  $$B(n) = 1$$
  - **worse case:**
  $$W(n) = \begin{cases} 1, & \text{if } n = 1 \text{ (base)} \\ 1 + W(n-1), & \text{otherwise (recursive call)} \end{cases}$$
  method: expand recursive call
  $$\begin{aligned} W(n) &= 1 + W(n-1) \\ &= 1 + (1 + W(n-2)) = 2 + W(n-2) \\ &= \dots \\ &= \underbrace{1 + 1 + \dots + 1}_{\text{k 1's}} + W(n-1) = k + W(n-k) \end{aligned}$$
  Now: use the base case
  $$\text{Let } n - k = 1$$
  $$\text{then } k = n - 1$$

$$W(n) = n - 1 + W(1)$$

$$= n - 1 + 1$$

$$= n$$

- **on average**:

$$A(n) \approx \frac{n}{2}$$

- **Space Complexity**
  - memory space for k, n, x, and s
  - stack space for n-1 cells
  - not efficient due to recursive calls
- **Optimality**
  - if not optimal =¿ show a better method
  - Sequential search is optimal for non-ordered array
  - Sequential search is NOT optimal for ordered array

3 Example 2: Binary Search

3.1 Algorithm description of binary search

- **Problem**: Find the index **k** of an element **x** in the ordered array **s** of **n** elements **s[1,n]**. If **x** is not in the array, return -1.
- **Inputs: s, L, U, x** where
  - s: array of n items to process
  - L: lower bound
  - U: upper bound
  - x: target to search for
- **outputs:** index k with s[k]=x or -1
- **Algorithm in Pseudocode**(Note that the operator we use to compute **m** is the **floor** operator.)

```
BS(s,L,U,x)
    if(L>U)
        return -1
    else
        m = ⌊(L+U)/2⌋
        if(x=s[m])
            return m
        else if (x¡s[m])
            BS(s,L,m-1,x)
        else if (x¡s[m])
            BS(s,m+1,U,x)
EndBS
```

3.2 Algorithm Analysis of Binary Search

In this part, we only focus on the time complexity.

- **Time complexity**
  - **best case**

$$B(n) = 1$$

  - **worst case**

$$W(n) = \begin{cases} 1, & \text{if } n = 1 \text{ (base)} \\ 1 + W(\left\lfloor \frac{n}{2} \right\rfloor), & \text{otherwise (recursive call)} \end{cases}$$

method: expand recursive call

$$\begin{aligned} W(n) &= 1 + W(\left\lfloor \frac{n}{2} \right\rfloor) \\ &= 1 + (1 + W(\left\lfloor \frac{\left\lfloor \frac{n}{2} \right\rfloor}{2} \right\rfloor)) = 2 + W(\left\lfloor \frac{n}{2^2} \right\rfloor) \\ &= \dots \\ &= k + W(\left\lfloor \frac{n}{2^k} \right\rfloor) \end{aligned}$$

Now: use the base case

$$\text{Let } \left\lfloor \frac{n}{2^k} \right\rfloor = 1$$

$$\text{then } 1 \leq \frac{n}{2^k} \leq 2$$

$$2^k \leq n \leq 2^{k+1}$$

$$k \leq log_2 n \leq k + 1$$

$$log_2 n - 1 \leq k \leq log_2 n$$

$$k = \lfloor log_2 n \rfloor$$

$$\begin{aligned} W(n) &= \lfloor log_2 n \rfloor + 1 \\ &= \lceil log_2 (n + 1) \rceil \end{aligned}$$