Lab 02

I. El-Shaarawy, & M. Touny

October 22, 2023

Contents

Pre-Lab Assumptions

- Students have attempted to reconstruct the arguments/examples given in the lecture.

Lab Goals

- Math preliminaries
  - Recurrences' telescoping (backward substitution)
  - Summation closed forms ($1+\cdots+n = \frac{n(n+1)}{2}$ closed form, and $\sum i+j = \sum i + \sum j$).
- Order rates of growth's complexity, by big O, big Omega, and big Theta notations.
- Given an algorithm: Decide size parameter, identify the basic operation, identify the worst-case, build a recurrence (or summation expression), and solve the recurrence, concluding the worst-case time complexity.

Post-lab Tasks

- Pick-up any algorithm we solved its time complexity, and modify the number of basic operations. Re-solve the whole analysis completely in your own words and justification without looking at the original solution.
- Given $n = 2^k$, Show the loop

  ```
  for (i=1; i<=n; i=i*2)
  ```

  takes exactly $\log(n)$ iterations. Use recurrences and back-substitution trick.

Exercises

2.1.1

(a)

```
def sumOfArrayNumbers(A[0 .. n-1])
  sum = 0
  for i in 0..n-1
    sum = sum + A[i]
  return sum
```

n; Summation; no.

(b)

```
def factorial(n)
  res = 1
```

```
  for i in n..1
    res = res * i
  return res
```

1; Multiplication; yes.

(c)

Algorithm is in page 61.

```
def maxElementInArray(A[0..n-1])
  max = A[0]
  for i in 1..n-1
    if A[i] > max
      max = A[i]
  return max
```

n; Comparison; no.

(d)

```
def gcd(m,n)
  while n != 0
    r = m mod n
    m = n
    n = r
  return m
```

2; mod operation; yes.

(e) Homework.

(f) Homework.

2.1.3

Yes.

Classical Search

- Worst-case is $\mathcal{O}(n)$.
- Average-case is $1 \cdot \frac{1}{n} + \cdots + n \cdot \frac{1}{n} = \frac{1}{n}(1 + \cdots + n) = \frac{1}{n}\frac{n(n+1)}{2} = \frac{n+1}{2} = \mathcal{O}(n)$.
- Best-case is $\mathcal{O}(1)$.

Varied Search

Worst-case, Average-case, and Best-case, are all $\Omega(n)$. For any determinstic algorithm not reading all the n cells, We can construct a counter-example input.

P.S. Big-Oh is used to upper-bound complexity, showing an algorithm is efficient. So it can't be used here while we are showing the inefficiency.

2.2.5

$5 \lg(n+100)^{10}, \ln^2 n, n^{1/3}, 0.001n^4 + 3n^2 + 1, 3^n, 2^{2n}, (n-2)!$

2.2.12

Homework

2.3.4

(a) $s(n)_{i=1}^{n} = \sum_{i=1}^{n} i * i$, The sum of squares up to n.

(b) Multiplication.

(c) $\sum_{i=1}^{n} 1 = n(1) = n$.

(d) $\mathcal{O}(n)$.

(e) Homework.

2.3.6

(a) Homework.

(b) Comparison.

(c) $\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-1) - (i+1) + 1 = \sum_{i=0}^{n-2} n - i - 1 = (n+1) + n + \cdots + (n - (n-2) - 1) = \frac{n(n+1)}{2}$

(d) $\mathcal{O}(n^2)$

(e) Homework.

2.4.3

(a)

$$\begin{aligned} S(n) &= S(n-1) + 2 \\ &= S(n-2) + 4 \\ &\cdots \\ &= S(n-(n-1)) + 2(n-1) \\ &= 0 + 2n - 2 \end{aligned}$$

Hence $\mathcal{O}(n)$.

(b) Homework.

2.4.9

(a) Homework.

(b)

$$\begin{aligned} S(n) &= S(n-1) + 1 \\ &= S(n-2) + 2 \\ &\dots \\ &= S(n-(n-1)) + n - 1 \\ &= 1 + n - 1 \end{aligned}$$

Hence $\mathcal{O}(n)$

2.5.10

Homework

2.5.12

Homework

2.6.1

No count of the comparison basic operation $A[j] > v$ in case it is $False$.

Fix by adding

```
if j >= 0 then count = count+1
```

after the end of while.

2.6.10

Homework