3 Analysis of Non-Recursive Functions

Tuesday, February 11, 2025 12:22 PM

E.G

```
FUNCTION MaxElement ( A[0... n-1] )
    max ← A[0]                         } 2
    FOR i ← 1 to n-1 DO
        IF A[i] > max THEN     } 2
            max ← A[i]         } 2
    RETURN max
```

$$\left.\begin{array}{l}\text{IF } A[i] > \max \text{ THEN } \} 2 \\ \max \leftarrow A[i] \ \} 2\end{array}\right\} 4 \qquad \sum_{i=1}^{n-1} 4 \qquad 2 + \sum_{i=1}^{n-1} 4$$

$$2 + (4 + 4 + 4 + 4 \ldots + 4)$$
$$i = 1 \quad 2 \quad 3 \quad 4 \ldots n\text{-}1$$

$$R_{MaxE}(n) = (n-1) \cdot 4 + 2$$
$$= 4n - 4 + 2$$
$$= 4n - 2 \in \Theta(n)$$

Computing whole runtime function: extra work 

Instead let us concentrate on a basic operation.

an operation that characterizes the algorithm

e.g. the comparison $>$

$$C_{MaxE}(n) = \sum_{i=1}^{n-1} 1 = n-1 \in \Theta(n)$$

E.G

```
FUNCTION Sum ( A[0..n-1] )
    s ← 0
    i ← 0
    FOR i ← 0 TO n-1 DO
        s ← s + A[i]
    return s
```

Analysis: Basic operation: $+$

$$S_{sum}(n) = \sum_{i=0}^{n-1} 1 = n \in \Theta(n)$$

GENERAL PLAN FOR ANALYSING THE EFFICIENCY OF NON-RECURSIVE ALGORITHMS

1) Decide on parameter(s) that indicate input size.

2) Identify the basic operation

   ★ find it in the innermost loop

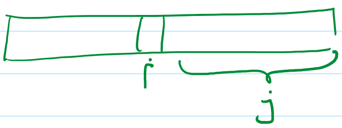
3) Check whether the number of times the basic operation is performed depends only on the input size.
   a. If so : worst-case analysis is enough.
   b. Otherwise : average-case or amortized analysis may be applicable.

4) Set up a sum counting the number of execution of the basic operation

5) Simplify the sum to closed form.
   If not possible, try to ascertain the order-of-growth of the function.

E.G.

```
FUNCTION allUnique ( A[0... n-1] )
  FOR i ← 0 TO n-2 DO
    FOR j ← i+1 TO n-1 DO
      IF A[i] = A[j] THEN
        RETURN False
  RETURN True
```



1) input-size n

2) basic operation: $=$

3) No, but still worst-case analysis

4) $$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

i j

5) $$C(n) = \sum_{i=0}^{n-2} [(n-1)-(i+1)+1]$$

$$= \sum_{i=0}^{n-2} [n-1-i]$$

$$= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i$$

$$= (n-1) \cdot \sum_{i=0}^{n-2} 1 - \sum_{i=0}^{n-2} i$$

$$= (n-1) \cdot (n-2-0+1) - \frac{(n-2)(n-1)}{2}$$

$$= (n-1)\cdot(n-2-0+1) - \frac{(n-2)(n-1)}{2}$$

$$= (n-1)^2 - \frac{(n-2)(n-1)}{2}$$

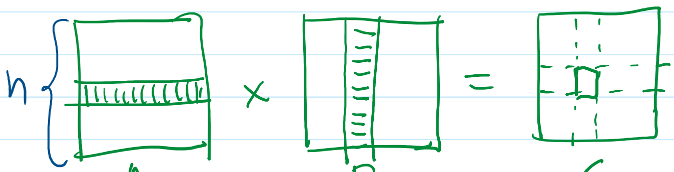
$$= n^2 - 2n + 1 - \frac{1}{2}n^2 - \frac{1}{2}n - \frac{1}{2}\cdot 2\cdot n + \frac{1}{2}\cdot 2$$

$$= \frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$$ "cuadratic"

- Note: Dont trust that a single loop is always n
  a double loop is always n^2

E.G. FUNCTION foo(A[0...n-1])
  FOR i = 0 TO n*n DO
    basic operation.

E.G. Matrix Multiplication



A B C

$$C[i,j] = A[i,0]\cdot B[0,j] + A[i,1]\cdot B[1,j] + A[i,2]\cdot B[2,j] + A[i,3]\cdot B[3,j]$$
$$\vdots$$
$$A[i,n-1]\cdot B[n-1,j]$$

```
FUNCTION SqMatrixMult( A, B, C )   (VAR C)
  FOR i <- 0 TO n-1 DO
    FOR j <- 0 TO n-1 DO
      C[i,j] <- 0.0
      FOR k <- 0 TO n-1 DO
        C[i,j] <- C[i,j] + A[i,k] * B[k,j]
```

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n$$

$$\sum_{i=0} \sum_{j=0} n$$

$$= \sum_{i=0}^{n-1} \left(n \cdot \sum_{j=0}^{n-1} 1 \right)$$

$$= \sum_{i=0}^{n-1} \left(n^2 \right)$$

$$= n^2 \cdot \sum_{i=0}^{n-1} 1$$

$$= n^3 \in \Theta(n^3)$$ "qubic"

- What about WHILE ?