3.1-2) By Θ definition, there must exists constants such that $c_1, c_2, n_0 > 0$ such that:

$$c_1 \cdot n^b \leq (n+a)^b \leq c_2 \cdot n^b, \quad \forall n > n_0$$

It is known that:

- $n+a \leq n+|a| \leq 2n$ if $n \geq |a|$
- $n+a \geq n-|a| \geq \frac{1}{2}n$ if $\frac{1}{2}n \geq |a| \Leftrightarrow n \geq 2|a|$

Out of these two, the inferior one is a tighter constraint. If it complies with the inferior constraint, it will also comply with the upper constraint.

Therefore,

$$\frac{1}{2}n \leq n+a \leq 2n, \quad \text{when } n \geq 2|a|$$

Since $b > 0$, the inequality also holds when all parts are raised to the b^{th} power:

$$\left(\frac{1}{2} \cdot n\right)^b \leq (n+a)^b \leq (2n)^b$$

$$\left(\frac{1}{2}\right)^b \cdot n^b \leq (n+a)^b \leq (2)^b \cdot n^b$$

Thus, $c_1 = \left(\frac{1}{2}\right)^b$

$c_2 = (2)^b$ satisfy the definition.

$n_0 = 2|a|$

4.) By defintion we have, for all $n \geq n_0$:

$$c_1 \cdot n^2 \leq 2n^2 + 3n + 1 \leq c_2 n^2$$

Divide by n^2

$$c_1 \leq 2 + \frac{3}{n} + \frac{1}{n^2} \leq c_2$$

Lets arbitrarily take $n_0 = 3$, then $2 + \frac{3}{3} + \frac{1}{9} = \frac{28}{9}$

Hence, choosing $0 < c_1 \leq \frac{28}{9}$ and $\frac{28}{9} \leq c_2 \leq +\infty$ for $n_0 = 3$.

. By definition, $2n^2 + 3n + 1 = \Theta(n^2)$

Algorithms

Pg 22   2041280366   Albert Sebastián Millán Trujillo

2.1) $A = \langle 31, 41, 59, 26, 41, 58 \rangle$

(a) | 31 | 41 | 59 | 26 | 41 | 58 |

(b) | 31 | 41 | 59 | 26 | 41 | 58 |

(c) | 31 | 41 | 59 | 26 | 41 | 58 |

(d) | 26 | 31 | 41 | 59 | 41 | 58 |

(e) | 26 | 31 | 41 | 41 | 59 | 58 |

(f) | 26 | 31 | 41 | 41 | 58 | 59 |

2.2) Insertion-Sort (Non-increasing)

```
for j=2 to A.length:
    key = A[j]
    i = j-1
    while i > 0 and A[i] < key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

2.3) Linear-Search (A, v)

```
res = NIL
i = 1
while i < (n+1) and res =
    if A[i] == v:
        res = i
    i = i+1

return res
```

Initialization: ~~case where A = <>. I~~
~~empty, hence it returns NIL.~~
Case where true before 1st itera
Because we haven't started the ite
there is no evidence that v
hence res = NIL.

Maintenance: res will hold NIL

Pg53

3.1-3) The statement uses big O notation, which essentially defines the upper bound. Hence saying 'at least' is meaningless since the lower bound is also the upper bound, and thus no running time can be greater than the upper bound.

3.1-1) According to the definition of $\Theta(\cdot)$, there must exist positive constants c_1, c_2 and n_0 such that

$$(1) \quad c_1 \cdot (f(n)+g(n)) \leq \max(f(n), g(n)) \leq c_2 \cdot (f(n)+g(n)), \quad \forall n > n_0$$

Since $f(n)$ and $g(n)$, ~~there exists n_0~~ are asymptotically non-negative, there exists an n_0 such that $f(n) \geq 0$ and $g(n) \geq 0$ for all $n \geq n_0$. It ~~does~~ thus follows that the following statements ~~is~~ are true:

$$\left.\begin{aligned} &\bullet\ f(n)+g(n) \geq f(n) \geq 0 \\ &\bullet\ f(n)+g(n) \geq g(n) \geq 0 \end{aligned}\right\} \forall n \geq n_0$$

Since $\max(f(n), g(n))$ can either be $f(n)$ or $g(n)$, the above statements already prove the right-most inequality for constant $c_2 = 1$. That is:

$$\max(f(n), g(n)) \leq 1 \cdot [f(n)+g(n)] \Rightarrow \begin{cases} f(n) \leq f(n)+g(n) \\ g(n) \leq f(n)+g(n) \end{cases}$$

Similarly the left-most inequality, we know that:

$$\begin{aligned} &\bullet\ 0 \leq f(n) \leq \max[f(n), g(n)] \\ &\bullet\ 0 \leq g(n) \leq \max[f(n), g(n)] \quad + \\ \hline &0 \leq f(n)+g(n) \leq 2 \cdot \max[f(n), g(n)] \Rightarrow \tfrac{1}{2} \cdot [f(n)+g(n)] \leq \max[f(n), g(n)] \end{aligned}$$

where $c_1 = 1/2$

Thus, we have found values for $c_1 = \frac{1}{2}$ and $c_2 = 1$ so that to comply with eq. 1. Hence, by definition, $\max(f(n), g(n)) = \Theta(f(n)+g(n))$

P29

2.2-2)

```
for i = 1 to (A.length - 1):
    small = i
    for j = i+1 to A.length:
        if A[small] > A[j]:
            small = j
    if small != i:
        swap(A[small], A[i])
```

Loop Invariants

There are two for loops so there are invariants for each:

- Outer loop
  - A[0,..., i-1] is in sorted ascending order
  - All entries from A[0,..., i-1] must be smaller or equal to the entries from A[i, n]
- Inner loop
  - All entries from A[i,..., j-1] should be larger or equal than A[small].

It only iterates up to n-1 because it is sorting in increasing order ~~the~~ the elements from A[0,..., i]. taking the minimum value from A[i,..., n] When i = n-1, all the elements up to n-1 are sorted ~~the~~ in increased order, taking the minimum value from A[n-1, n], hence, the remaining value at position n must be greater than that at n-1, and thus all that preceed it.

Performance

Best Case: $\Theta(n^2)$

Worst Case: $\Theta(n^2)$

The dominant terms are the two for loops. It performs the same number of operations in both the best and worst cases as it compares element-by-element.

2.2-3) On the average case, there need to be $(n/2)$ elements checked.

It is the point in-between the upper (n) ~~bound~~ and the lower (1) bound.

In the worst case, the method iterates over all the elements in the array, hence 'n' times

Average ~~Best~~ Case: $\Theta(n)$

Worst Case: $\Theta(n)$

This is because the leading coefficient is ignored in theta (Θ) notation.

Pg 22

2.3) Continued

Code

```
for i = 1 to A.length
    if A[i] == v
        return i
return NIL
```

Loop Invariant:

$\forall k \in [1, i),\ A[k] \neq v$

Initialization:

$i == 1 \Rightarrow [1, i) == \emptyset \Rightarrow \forall k \in \emptyset,$
$A[k] \neq v$

Maintenance: Lets assume true for the first $i-1$ iterations of the for loop.
At the start of the i^{th} iteration, if $A[i] = v$, the current iteration is the final one (see termination). Otherwise, if $A[i] \neq v$, we have:

$\forall k \in [1, i),\ A[k] \neq v$ and $A[i] \neq v \iff \forall k \in [1, i+1),\ A[k] \neq v$,

which means the invariant loop will also be true at the start of the next iteration ($(i+1)^{th}$).

Termination: the for loop may end for two reasons.

1. If $A[i] == v$, it will return the value of i. this case satisfies the loop invariant the values of A in previous iterations is such that $0, \dots, (i-1)^{th}$ are not equal to v.
$A[i] == v \Rightarrow \forall k \in [1, i),\ A[k] \neq v$

2. If $i == A.length + 1$, in which case we are at the beginning of the $(A.length+1)^{th}$ iteration
$\forall k \in [1, A.length+1),\ A[k] \neq v \iff \forall k \in [1, A.length],\ A[k] \neq v$
and NIL is the value returned. Hence, the conditions remain true.