ex: El cost de l'algorisme de selecció és $\Theta(n^2)$

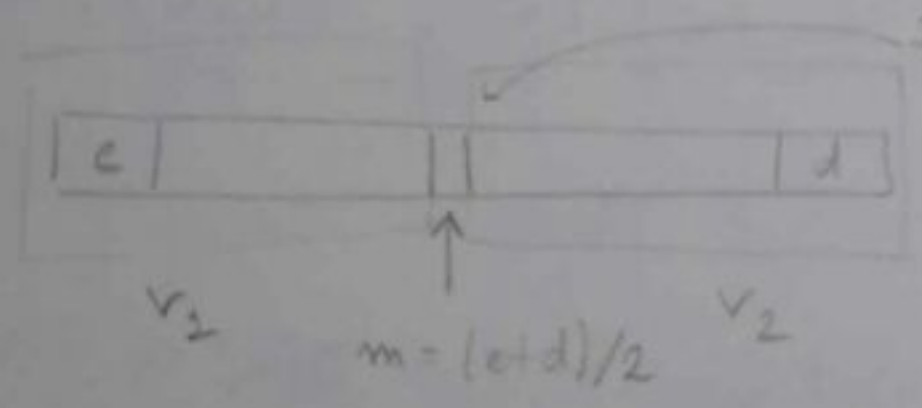
inserció en el cas millor és $\Theta(n)$

i en el cas pitjor és $\Theta(n^2)$

Cerca dicotòmica

Input: Vector ordenat v; element x

Output: Determinar si x és o no a v



si $x == v[m]$ → ja estem

sino si $x < v[m]$

return cerca x a v_1 → fem la meitat amb la primera meitat → recursiu

si $x > v[m]$ (o sino)

↳ fem la meitat amb la segona meitat

return cerca x a v_2

cas millor: $\Theta(1)$ <--> "és constant"

cas pitjor $\Theta(\log(n))$

2/10 d'Una

1) Regles per calcular el cost dels algorismes iteratius:

* El cost de qualsevol operació elemental és $\Theta(1)$

* Composició seqüencial Donats dos fragments de codi S_1 i S_2 amb costos f_1 i f_2 respectivament, el cost del fragment S_1; S_2,

és $f_1 + f_2$

* Composició alternativa (condicional) S_1, S_2 amb costos f_1 i f_2 ; el cost d'avaluar A f_c, llavors el cost en el pitjor del fragment

```
if (A) {
    S1;
}
else {
    S2;
}
```

és $f_a + \max\{f_1, f_2\}$

* Composició iterativa.

Sigui A_i una condició a avaluar a la i-èsima iteració, S_i el fragment de codi a executar a la i-èsima iteració i f_{a_i}, f_i els costos respectius

$f_a = \max\{f_{a_i}\}$

$f = \max\{f_i\}$

Llavors el cost en cas pitjor del fragment, amb n iteracions és

```
while (A) {
    Si
}
```

$\Theta(n \cdot (f_a + f))$

<u>exemples</u>

1)
```
int s = 0
for (int i = 1; i < n; i *= 2) {
    ++s;
}
```

$\Theta(1) + \Theta(\log(n)) = \Theta(\log(n))$

2)
```
int s = 0
for (int i = 0; i < n; ++i) {      ← s'hauria n vegades algo
    for (int j = 0; j < i; ++j) {
        ++s
    }                                ← en el pitjor cas i = n-1 i serà Θ(n)
}
```

$\Theta(1) + n \cdot \Theta(n) = \Theta(n^2)$

3)
```
int s=0;
for (int i=0; i<n; ++i){
    for (int j=0, j<i*i; ++j){
        if (j%i == 0){
            for (int k=0, k<n; ++k){
                ++s
            }
        }
    }
}
```

$$\sum_{i=0}^{n}\left(\sum_{j=0}^{i^2} \Theta(1) + i\cdot\Theta(n)\right) = \Theta\left(\sum_{i=0}^{n} i^2\right) + \Theta\left(n\cdot\sum_{i=0}^{n} i\right)$$

$$= \Theta(n^3) + \Theta(n^3) = \Theta(n^3)$$

2) Càlcul del cost dels algorismes recursius

→ Recurrències substractores

$$T(n) = \begin{cases} \Theta(1) & n \leq c \\ a\,T(n-c) + g(n) & n > c \end{cases}$$

← o cas base

$c > 0$
$g = \Theta(n^k)$

Teorema fonamental

$$T(n) = \begin{cases} \Theta(n^k) & a < 1 \\ \Theta(n^{k+1}) & a = 1 \\ \Theta(a^{n/c}) & a > 1 \end{cases}$$

```
int fib(n){
    if(n==0) return 0;
    if(n==1) return 1;
    return fib(n-1) + fib(n-2);
}
```

$$F(n) = \begin{cases} O(1) & n < 2 \\ & n \geq 2 \end{cases}$$

$$2F(n-2) \leq F(n) = F(n-1) + F(n-2) \leq 2F(n-1)$$

$F(n) = O(2^n)$ $\qquad$ $F(n) = \Omega(2^{n/2})$

$\longrightarrow$ Recorrències divisores

$$T(n) = \begin{cases} \Theta(1) & n < b \\ a\,T\left(\frac{n}{b}\right) + g(n) & n \geq c \end{cases}$$

$b > 1$

$g = \Theta(n^k)$, $\alpha = \log_b(a)$

$$T(n) = \begin{cases} \Theta(n^k) & \alpha < k \\ \Theta(n^k \cdot \log(n)) & \alpha = k \\ \Theta(n^\alpha) & \alpha > k \end{cases}$$

Teorema

Exemples:

- cerca dicotòmica: $T(n) = \begin{cases} \Theta(1) & n \leq 1 \\ T\left(\frac{n}{2}\right) + \Theta(1) & n > 1 \end{cases}$

$\alpha = \log_2(1) = 0$, $k = 0$ $\qquad$ $T(n) = \Theta(\log(n))$

- exponenciació ràpida

$$x^n = \begin{cases} 1 & n = 0 \\ x^{(\lfloor n/2 \rfloor)^2} & \text{si } n \text{ és parell} \\ x^{(\lfloor n/2 \rfloor)^2} \cdot x & \text{si } n \text{ és senar} \end{cases}$$

```
double exprapida1(double x, int n) {
    if (n == 0) return 1;
    double y = exprapida(x, n/2);
    if (n%2 == 0) return exprapida1(x, n/2) * exprapida1(x, n/2);
                  return exprapida1(x, n/2) * exprapida(x, n/2) * x;
}
```

$$T(n) = 2\,T\left(\frac{n}{2}\right) + \Theta(1)$$

$$\alpha = 1,\ k = 0 \Rightarrow T(n) = \Theta(n)$$

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$$

$$\alpha = 0,\ k = 0 \Rightarrow T(n) = \Theta(\log n)$$