Taller de busqueda (searching)

Laboratorio
Algoritmos y Estructura de Datos I

DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

1er Cuatrimestre 2019

Problemas “difíciles”

Unos de los problemas difíciles más comunes son los de **búsqueda** (searching) y **ordenamiento** (sorting)

Problemas “difíciles”

Unos de los problemas difíciles más comunes son los de **búsqueda** (searching) y **ordenamiento** (sorting)

Lo difícil es pensar algoritmos “eficientes”

Algoritmos “eficientes”

Un algoritmo eficiente es el que resulve el problema con la menor cantidad de operaciones posible.

Algoritmos “eficientes”

Un algoritmo eficiente es el que resulve el problema con la menor cantidad de operaciones posible.

Queremos **saber de qué depende principlmente**
la cantidad de operaciones que realiza

Algoritmos “eficientes”

Un algoritmo eficiente es el que resulve el problema con la menor cantidad de operaciones posible.

Queremos **saber de qué depende principlmente** la cantidad de operaciones que realiza

Ejemplo:

- Un algoritmo con una entrada de tamaño n

$$\underbrace{k_1\, n \log(n) + k_3}_{\substack{\text{Cantidad exactas de} \\ \text{operaciones peor caso}}} = \underbrace{O(n \log(n))}_{\substack{\text{Complejidad} \\ \text{peor caso}}}$$

Problema de búsqueda

Muchas veces necesitamos saber si un elemento está en una lista:

proc $buscar(\text{in } s : seq\langle\mathbb{Z}\rangle, \text{in } x : \mathbb{Z}, \text{out } res : \text{Bool})\{$
  Pre $\{True\}$
  Post $\{res = true \leftrightarrow (\exists i : \mathbb{Z})(0 \leq i < |s| \wedge_L s[i] = x)\}$
$\}$

Listas desordenadas

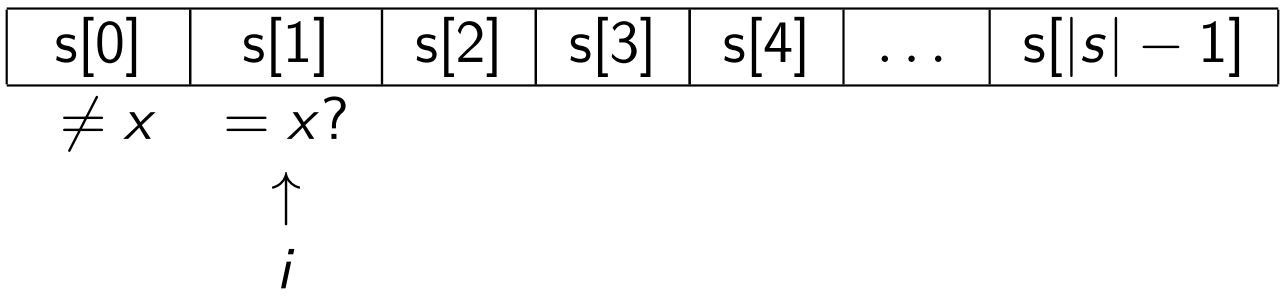
- Problema: buscar un elemento en una lista no ordenada.

Listas desordenadas

- Problema: buscar un elemento en una lista no ordenada.
- Algoritmo: búsqueda lineal. Recorre en orden la lista.

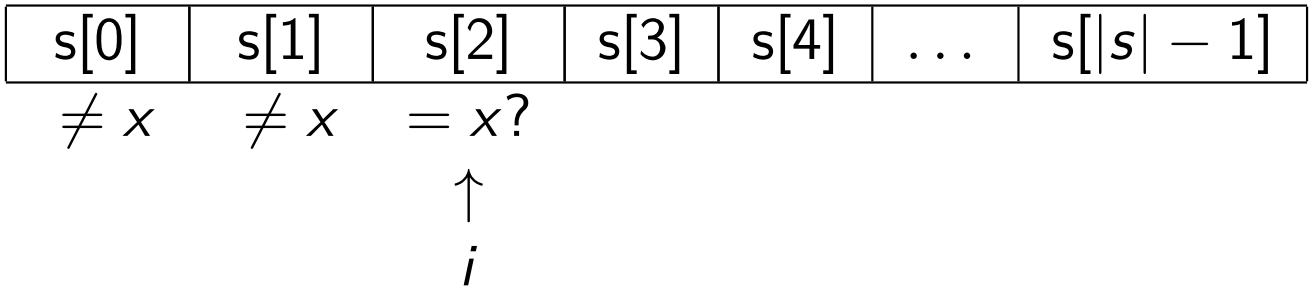
Listas desordenadas

- Problema: buscar un elemento en una lista no ordenada.
- Algoritmo: búsqueda lineal. Recorre en orden la lista.



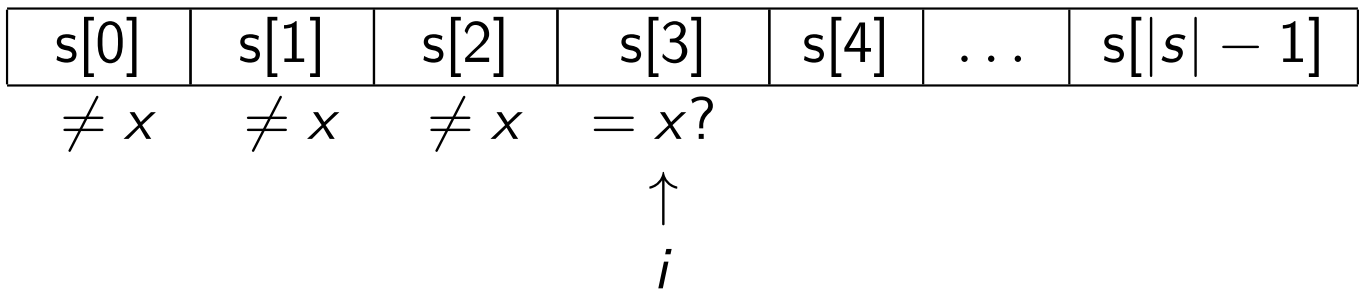
Listas desordenadas

- Problema: buscar un elemento en una lista no ordenada.
- Algoritmo: búsqueda lineal. Recorre en orden la lista.



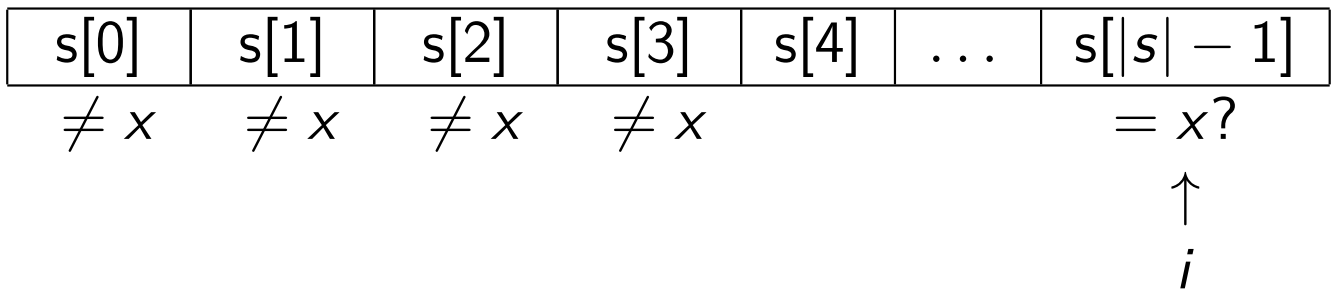
Listas desordenadas

- Problema: buscar un elemento en una lista no ordenada.
- Algoritmo: búsqueda lineal. Recorre en orden la lista.



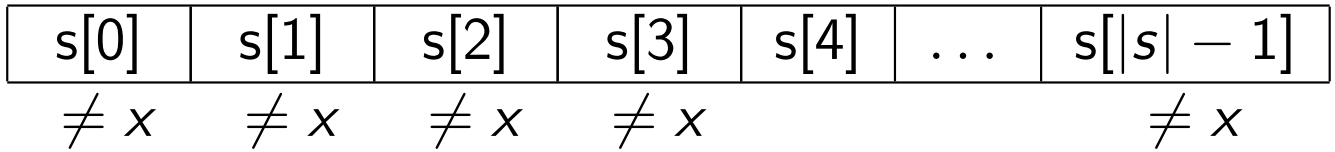
Listas desordenadas

- Problema: buscar un elemento en una lista no ordenada.
- Algoritmo: búsqueda lineal. Recorre en orden la lista.



Listas desordenadas

- Problema: buscar un elemento en una lista no ordenada.
- Algoritmo: búsqueda lineal. Recorre en orden la lista.



Complejidad $O(|s|)$

La cantidad de operaciones depende de la longitud de la lista.

Búsqueda lineal

```
1  bool busquedaLineal(vector<int> lista, int elem){
2    bool res = false;
3    for(int i=0; i < lista.size() ; i++){
4      if(lista[i]==elem){
5        res = true;
6      }
7    }
8    return res;
9  }
```

Listas ordenadas

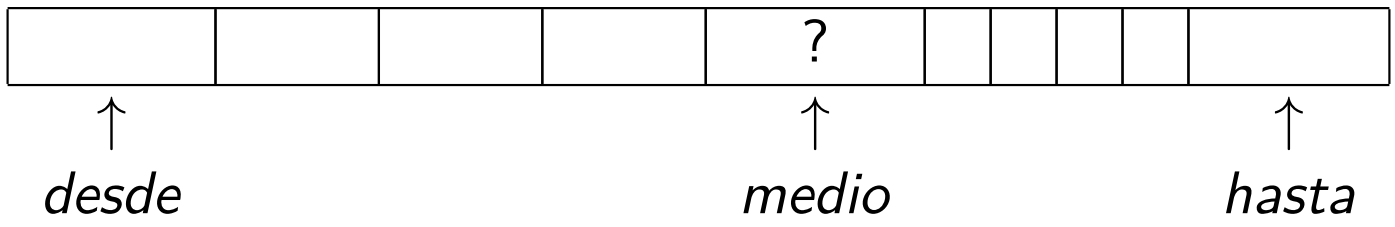
- Problema: buscar un elemento en una lista ordenada.

Listas ordenadas

- Problema: buscar un elemento en una lista ordenada.
- Algoritmo: Búsqueda binaria. Mirar el del medio. Si es mayor, seguir por la derecha, sino izquierda.

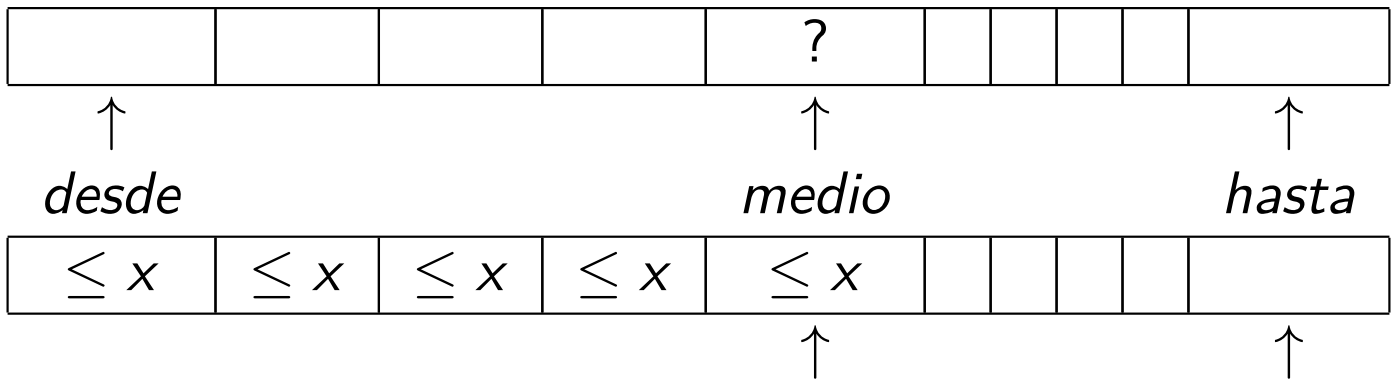
Listas ordenadas

- Problema: buscar un elemento en una lista ordenada.
- Algoritmo: Búsqueda binaria. Mirar el del medio. Si es mayor, seguir por la derecha, sino izquierda.



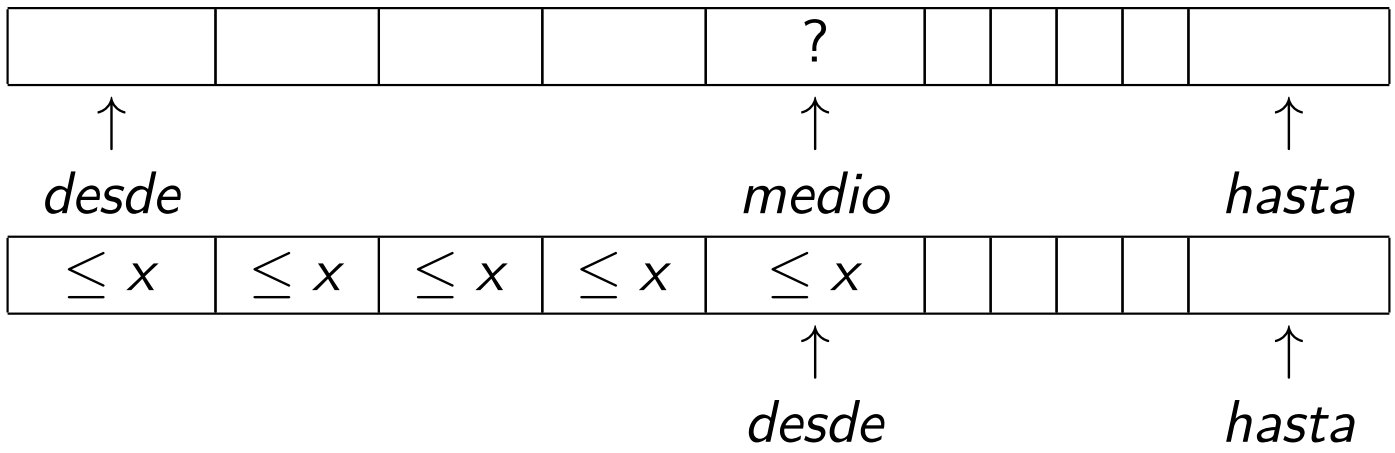
Listas ordenadas

- Problema: buscar un elemento en una lista ordenada.
- Algoritmo: Búsqueda binaria. Mirar el del medio. Si es mayor, seguir por la derecha, sino izquierda.



Listas ordenadas

- Problema: buscar un elemento en una lista ordenada.
- Algoritmo: Búsqueda binaria. Mirar el del medio. Si es mayor, seguir por la derecha, sino izquierda.



Búsqueda binaria

```
1  bool busquedaBin(vector<int> lista, int desde, int hasta, int e){
2    bool res = false;
3    while (desde <= hasta){
4      int medio = desde + (hasta-desde)/2;
5      if (lista[medio] == e){
6        res = true; //encontre el elemento
7      }
8      if (lista[medio] < e){
9        desde = medio + 1; //esta en la mitad derecha
10     }else{
11       hasta = medio - 1; //esta en la mitad izquierda
12     }
13   }
14   return res;
15 }
```

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia $hasta - desde$
0	$\lvert s \rvert - 1$

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia *hasta* − *desde*
0	$\lvert s\rvert - 1$
1	$\cong (\lvert s\rvert - 1)/2$

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia $hasta - desde$
0	$\lvert s\rvert - 1$
1	$\cong (\lvert s\rvert - 1)/2$
2	$\cong (\lvert s\rvert - 1)/4$

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia *hasta* − *desde*
0	$\lvert s\rvert - 1$
1	$\cong (\lvert s\rvert - 1)/2$
2	$\cong (\lvert s\rvert - 1)/4$
3	$\cong (\lvert s\rvert - 1)/8$

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia *hasta* − *desde*
0	$\lvert s\rvert - 1$
1	$\cong (\lvert s\rvert - 1)/2$
2	$\cong (\lvert s\rvert - 1)/4$
3	$\cong (\lvert s\rvert - 1)/8$
$\vdots$	$\vdots$

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia *hasta* − *desde*
0	$\lvert s\rvert - 1$
1	$\cong (\lvert s\rvert - 1)/2$
2	$\cong (\lvert s\rvert - 1)/4$
3	$\cong (\lvert s\rvert - 1)/8$
$\vdots$	$\vdots$
t	$\cong (\lvert s\rvert - 1)/2^t$

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia $hasta - desde$
0	$\lvert s\rvert - 1$
1	$\cong (\lvert s\rvert - 1)/2$
2	$\cong (\lvert s\rvert - 1)/4$
3	$\cong (\lvert s\rvert - 1)/8$
$\vdots$	$\vdots$
t	$\cong (\lvert s\rvert - 1)/2^t$

- Sea t la cantidad de iteraciones necesarias para llegar a $high - low = 1$.

Complejidad busqueda binaria

- ¿Cuántas iteraciones realiza el ciclo (en peor caso)?

Número de iteración	Subsecuencia *hasta* − *desde*
0	$\lvert s\rvert - 1$
1	$\cong (\lvert s\rvert - 1)/2$
2	$\cong (\lvert s\rvert - 1)/4$
3	$\cong (\lvert s\rvert - 1)/8$
$\vdots$	$\vdots$
t	$\cong (\lvert s\rvert - 1)/2^t$

- Sea t la cantidad de iteraciones necesarias para llegar a $high - low = 1$.

$$1 \cong (|s| - 1)/2^t \implies 2^t \cong |s| - 1 \implies t \cong \log_2(|s| - 1)$$

¿Si ir para atras fuera muuuy costoso?

Jump Search

Descripción:

- Empezamos con saltos de tamaño fijo, m.
- Cuando nos pasamos, hacemos búsqueda lineal.

Jump Search

Descripción:

- Empezamos con saltos de tamaño fijo, m.
- Cuando nos pasamos, hacemos búsqueda lineal.

Ejemplo

Buscar el elemento x=10 de a pasos de tamaño $m = 3$

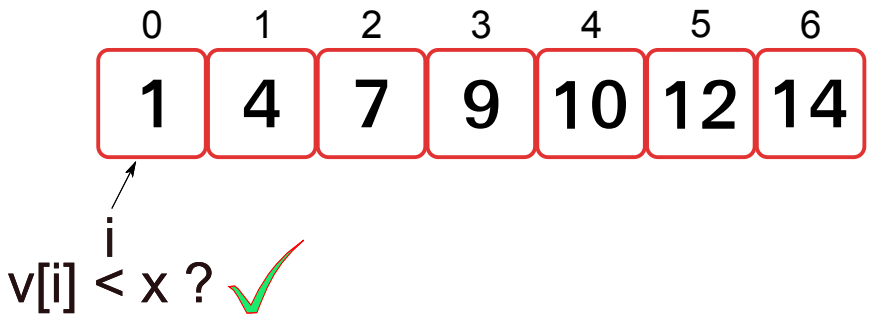
Jump Search

Descripción:

- Empezamos con saltos de tamaño fijo, m.
- Cuando nos pasamos, hacemos búsqueda lineal.

Ejemplo

Buscar el elemento x=10 de a pasos de tamaño $m = 3$



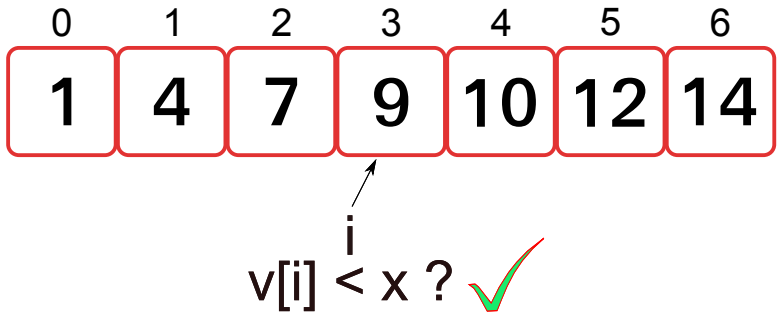
Jump Search

Descripción:

- Empezamos con saltos de tamaño fijo, m.
- Cuando nos pasamos, hacemos búsqueda lineal.

Ejemplo

Buscar el elemento x=10 de a pasos de tamaño $m = 3$



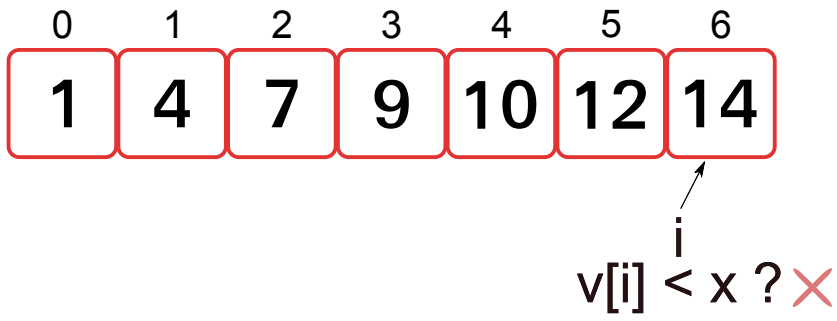
Jump Search

Descripción:

- Empezamos con saltos de tamaño fijo, m.
- Cuando nos pasamos, hacemos búsqueda lineal.

Ejemplo

Buscar el elemento x=10 de a pasos de tamaño $m = 3$



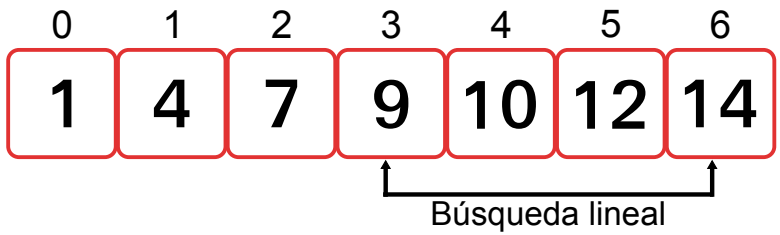
Jump Search

Descripción:

- Empezamos con saltos de tamaño fijo, m.
- Cuando nos pasamos, hacemos búsqueda lineal.

Ejemplo

Buscar el elemento x=10 de a pasos de tamaño $m = 3$



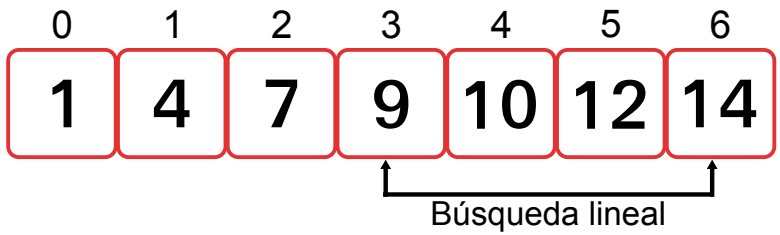
Jump Search

Descripción:

- Empezamos con saltos de tamaño fijo, m.
- Cuando nos pasamos, hacemos búsqueda lineal.

Ejemplo

Buscar el elemento x=10 de a pasos de tamaño $m = 3$



Código

Ustedes!

Iteraciones

¿Cuántas iteraciones hace en el peor caso?

Iteraciones

¿Cuántas iteraciones hace en el peor caso?

- En el peor caso, hacemos $\frac{|s|}{m}$ saltos
- Al final, tenemos $m - 1$ pasos más por la búsqueda lineal.

Iteraciones

¿Cuántas iteraciones hace en el peor caso?

- En el peor caso, hacemos $\frac{|s|}{m}$ saltos
- Al final, tenemos $m - 1$ pasos más por la búsqueda lineal.

$$\text{iteraciones} \approx \frac{|s|}{m} + m - 1$$

Complejidad

Sabiendo que el bloque óptimo es $m = \sqrt{|s|}$,

¿Cuál es la complejidad?

Complejidad

Sabiendo que el bloque óptimo es $m = \sqrt{|s|}$,

¿Cuál es la complejidad?

$$O(\frac{|s|}{\sqrt{|s|}} + \sqrt{|s|} - 1)$$

Complejidad

Sabiendo que el bloque óptimo es $m = \sqrt{|s|}$,

¿Cuál es la complejidad?

$$O(\frac{|s|}{\sqrt{|s|}} + \sqrt{|s|} - 1) = O(\frac{\sqrt{|s|}}{\cancel{\sqrt{|s|}}} \frac{\cancel{|s|}}{\cancel{\sqrt{|s|}}} + \sqrt{|s|} - 1)$$

Complejidad

Sabiendo que el bloque óptimo es $m = \sqrt{|s|}$,

¿Cuál es la complejidad?

$$O(\frac{|s|}{\sqrt{|s|}} + \sqrt{|s|} - 1) = O(\frac{\sqrt{|s|}}{\cancel{\sqrt{|s|}}} \frac{\cancel{|s|}}{\cancel{\sqrt{|s|}}} + \sqrt{|s|} - 1)$$
$$= O(\sqrt{|s|} + \sqrt{|s|} - 1)$$

Complejidad

Sabiendo que el bloque óptimo es $m = \sqrt{|s|}$,

¿Cuál es la complejidad?

$$
\begin{aligned}
O(\frac{|s|}{\sqrt{|s|}} + \sqrt{|s|} - 1) &= O(\frac{\sqrt{|s|}}{\cancel{\sqrt{|s|}}} \frac{\cancel{|s|}}{\cancel{\sqrt{|s|}}} + \sqrt{|s|} - 1) \\
&= O(\sqrt{|s|} + \sqrt{|s|} - 1) \\
&= O(2\sqrt{|s|} - 1)
\end{aligned}
$$

Complejidad

Sabiendo que el bloque óptimo es $m = \sqrt{|s|}$,

¿Cuál es la complejidad?

$$
\begin{aligned}
O(\frac{|s|}{\sqrt{|s|}} + \sqrt{|s|} - 1) &= O(\frac{\sqrt{|s|}}{\cancel{\sqrt{|s|}}} \frac{\cancel{|s|}}{\cancel{\sqrt{|s|}}} + \sqrt{|s|} - 1) \\
&= O(\sqrt{|s|} + \sqrt{|s|} - 1) \\
&= O(2\sqrt{|s|} - 1) \\
&= O(\sqrt{|s|})
\end{aligned}
$$