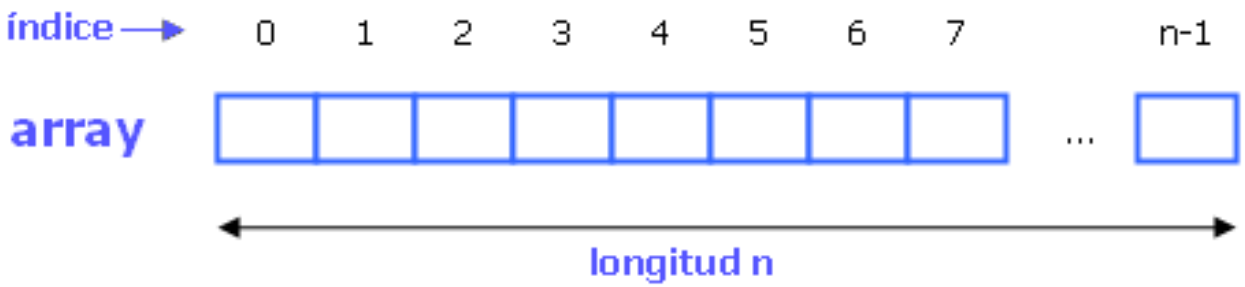
Estructura I

Autor: Cristian Jeldes

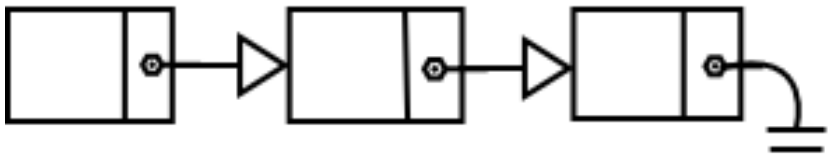
Estructuras básicas

Arreglo



Insertar: Constante
Buscar : N
Actualizar: Constante
Sacar: Constante
Agrandar: N

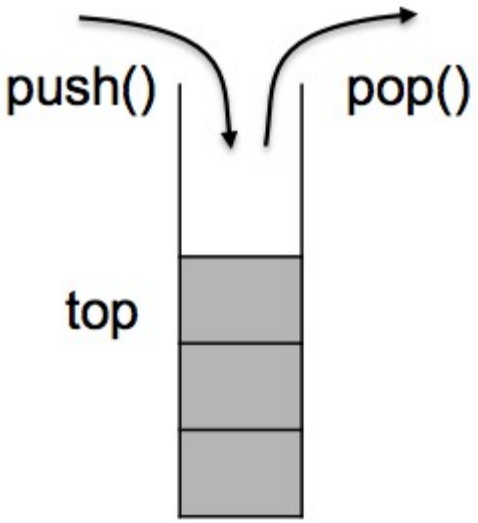
Listas



Lista enlazada

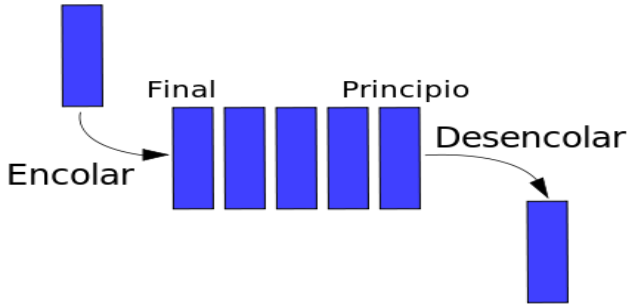
Insertar: Constante
Buscar : N
Actualizar: N
Sacar: N
Agrandar: Constante

Pila



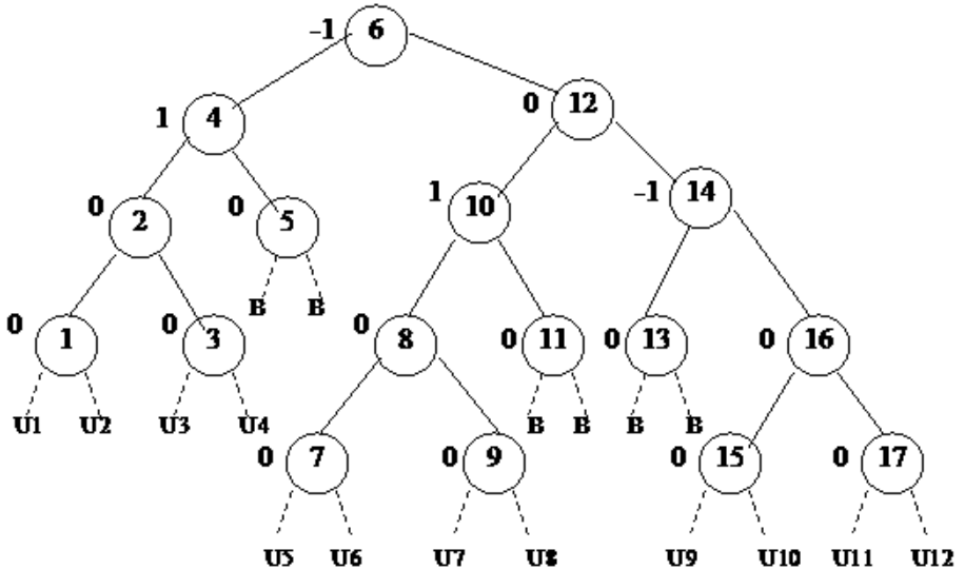
Insertar: Constante
Buscar : N
Actualizar: N
Sacar: Constante
Agrandar: Constante

Cola



Insertar: Constante
Buscar : N
Actualizar: N
Sacar: Constante
Agrandar: N

Árbol Binario Balanceado

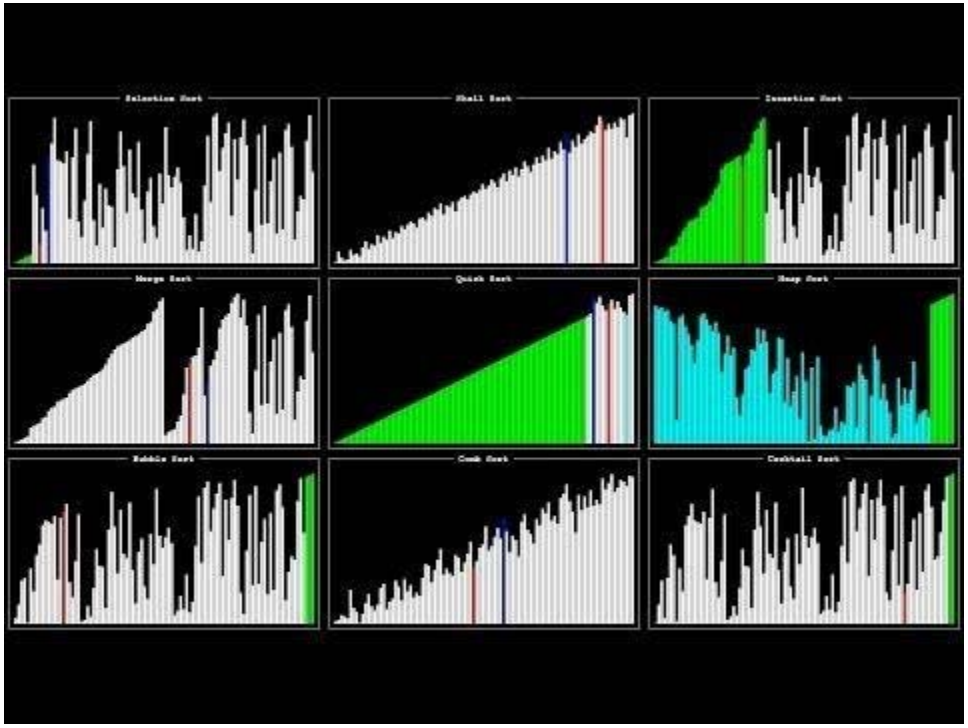


Insertar: Log(N)
Buscar : Log(N)
Actualizar: Log(N)
Sacar: Log(N)
Agrandar: Constante

Algoritmo de Ordenamiento

QuickSort

(https://www.youtube.com/watch?v=tIYMCYooo3c)



QuickSort

```python
def quicksort(L, first, last):
    # definimos los índices y calculamos el pivote
    i = first
    j = last
    pivote = (L[i] + L[j]) / 2

    # iteramos hasta que i no sea menor que j
    while i < j:
        # iteramos mientras que el valor de L[i] sea menor que pivote
        while L[i] < pivote:
            # Incrementamos el índice
            i+=1
        # iteramos mientras que el valor de L[j] sea mayor que pivote
        while L[j] > pivote:
            # decrementamos el índice
            j-=1
        # si i es menor o igual que j significa que los índices se han cruzado
        if i < = j:
            # intercambiamos los valores de L[j] y L[i]
            swap(L[i],L[j])
            # incrementamos y decrementamos i y j respectivamente
            i+=1
            j-=1

    # si first es menor que j mantenemos la recursividad
    if first < j:
        L = quicksort(L, first, j)
    # si last es mayor que i mantenemos la recursividad
    if last > i:
        L = quicksort(L, i, last)

    # devolvemos la lista ordenada
    return L
```

Algoritmos de Búsqueda

Lineal

Complejidad: O(N*Q)

N = Número de elementos

Q = Querys/Consultas

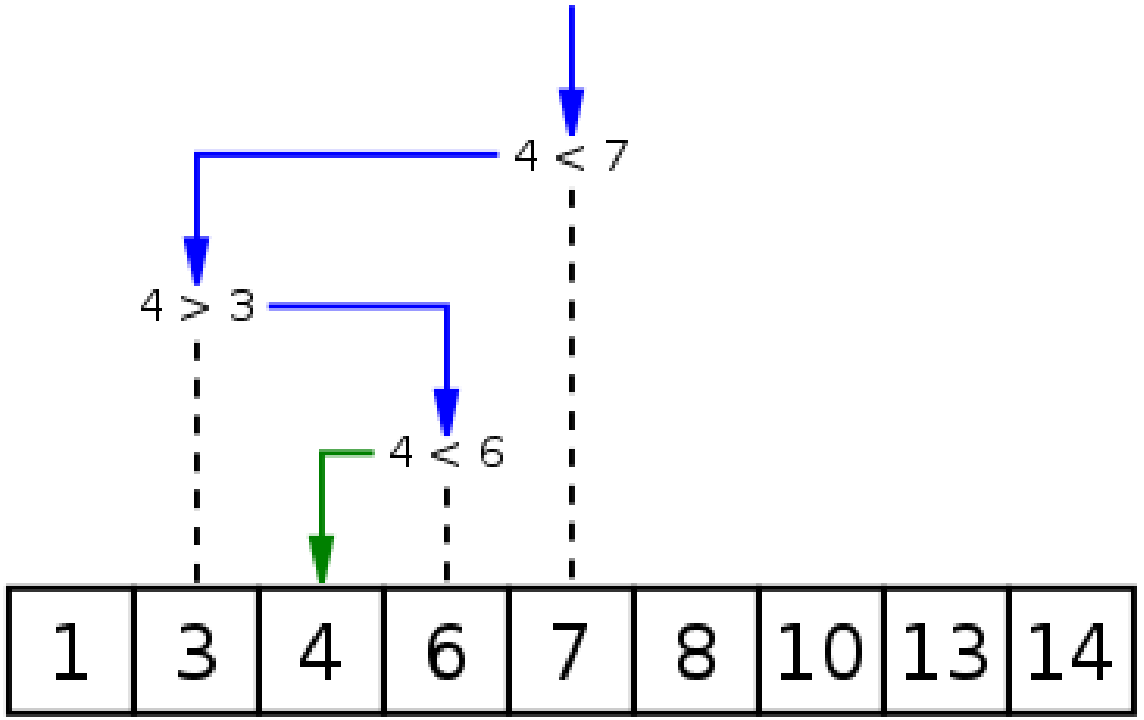
Binary Search

Complejidad: O(Log(N)*Q)

N = Número de elementos

Q = Querys/Consultas

Binary Search



Binary Search

```
int busquedaBinaria(int  vector[], int n, int dato)
{
   int centro,inf=0,sup=n-1;

   while(inf<=sup){
     centro=(sup+inf)/2;
     if(vector[centro]==dato) return centro;
     else if(dato < vector [centro] ){
        sup=centro-1;
     }
     else {
       inf=centro+1;
     }
   }
   return -1;
}
```

Problema de búsqueda en espacios numéricos

Tengo una función lineal creciente y necesito encontrar un valor v tal que al evaluar f(x) en v, el resultado sea k.

¿Cómo lo puedo hacer?

Exponential Search

Complejidad: O(Log(N)*Q)

N = Número de elementos

Q = Querys/Consultas

Exponential Search

F(x) = x/2, k=896

x	f(x)	> k?	< k ?	Fin?
1	0.5		Si	
2	1		Si	
4	2		Si	
8	4		Si	
16	8		Si	
32	16		Si	
64	32		Si	
128	64		Si	
256	128		Si	
512	256		Si	
1024	512		Si	
2048	1024	Si		
1536	768		Si	
1792	896			Si

Problema adicional

Tengo 2 celulares muy resistentes y un edificio de 100 pisos, ¿Cuál es el piso del que si tiro un celular, este se rompe?

Aplicaciones de la búsqueda binaria

Ejercicio de búsqueda binaria

13	21	25	33	34
16	***21***	***33***	***35***	35
16	***33***	***33***	***45***	50
23	***51***	***66***	***83***	93

interval=[20,90]

13	21	25	***33***	***34***
16	21	33	***35***	***35***
16	33	33	45	50
23	51	66	83	93

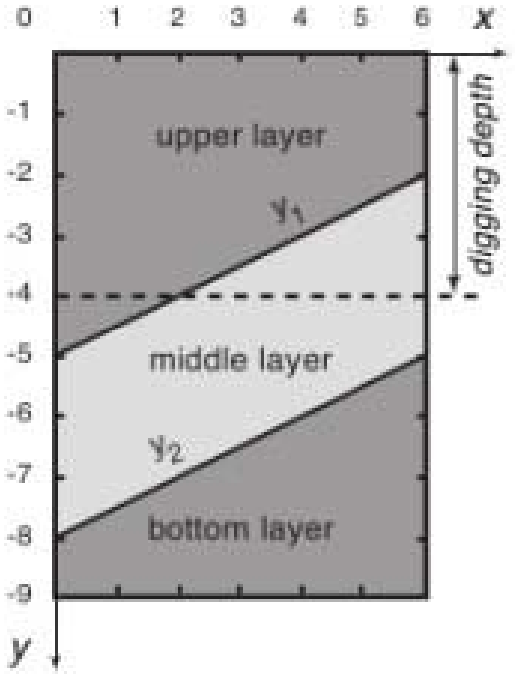
interval=[33,35]

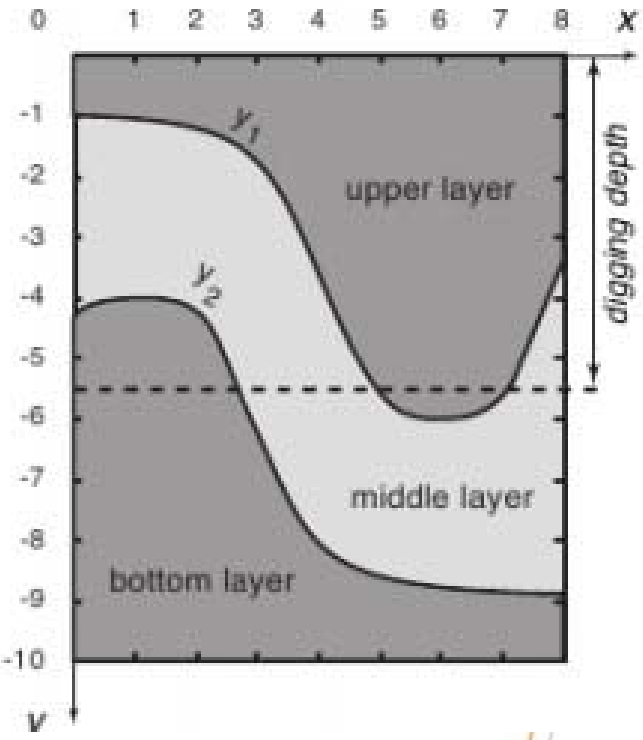
13	***21***	***25***	***33***	***34***
16	***21***	***33***	***35***	***35***
16	***33***	***33***	***45***	***50***
23	***51***	***66***	***83***	***93***

interval=[20,100]

https://icpcarchive.ecs.baylor.edu/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=2479

Aproximar un resultado





Tip: Iterar entre 20~50 veces como máximo.
Sino TLE.
Similar:
https://icpcarchive.ecs.baylor.edu/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=4038