Algoritmos de Busqueda:

1. Busqueda Binaria

* Como Funciona

Para poder realizar la búsqueda binaria, es necesario que el arreglo este ordenado. Primero se determina la mitad del arreglo, se compara el valor almacenado en la mitad con el valor buscado, si no coinciden y el valor buscado es mayor, se vuelve a determinar la mitad del arreglo en la parte superior. Si el valor buscado hubiera sido menor, se determinaría la mitad del arreglo en la parte inferior.

* Ventajas

Es el método mas eficiente para encontrar elementos en un arreglo ordenado.

* Aplicaciones

Se utiliza para la verificación y búsqueda de pagos.

* Pseudocodigo
* Procedure binary\_search
* A ← sorted array
* n ← size of array
* x ← value to be searched
* Set lowerBound = 1
* Set upperBound = n
* while x not found
* if upperBound < lowerBound
* EXIT: x does not exists.
* set midPoint = lowerBound + ( upperBound - lowerBound ) / 2
* if A[midPoint] < x
* set lowerBound = midPoint + 1
* if A[midPoint] > x
* set upperBound = midPoint - 1
* if A[midPoint] = x
* EXIT: x found at location midPoint
* end while
* end procedure

1. Busqueda Secuencial

* Como Funciona

La búsqueda secuencial compara los valores uno a uno del arreglo hasta recórrelo por completo y verifica si el valor buscado se encuentra en el arreglo. La ventaja de la búsqueda secuencial es que el arreglo no necesita estar ordenado.

* Ventajas

Es un método simple que es útil cuando se tiene un conjunto de datos pequeños.

Es el único método que puede usarse si el arreglo no está ordenado.

* Aplicaciones

Se utiliza en programas donde no se maneja una gran cantidad de datos, por ejemplo, se ha utilizado en aplicaciones en las que se necesita encontrar a una persona y la cantidad de personas es menor a 20.

* Pseudocodigo
* Programa: Búsqueda secuencial
* Variables
* T=10:entero
* a[T],temp,i,j,n:entero
* x:binario
* Inicio
* escribir "Llenando arreglo con números aleatorios"
* desde i=0 hasta i< T incremento 1 hacer
* a[i]=númeroaleatorio
* fin desde
* escribir "Numero a buscar? "
* leer n
* x=falso
* desde i=0 hasta i< T incremento 1 hacer
* si a[i] = n entonces
* escribir "Valor encontrado"
* escribir "Posición:”, i
* x=verdadero
* fin si
* fin desde
* si x=falso entonces
* escribir “No se encontró el el número”
* fin si
* escribir “El arreglo era:"
* desde i=0 hasta i< T incremento 1 hacer
* escribir a[i]  
   fin desde
* Fin

Algoritmos de Ordenamiento:

1. BubbleSort

* Como funciona

El algoritmo va revisando cada elemento del arreglo y lo compara con el siguiente elemento del arreglo, los intercambia de posición si están en el orden equivocado, es necesario revisar varias veces hasta que todo el arreglo este ordenado.

* Ventajas

No requiere memoria adicional a comparación de otros algoritmos de ordenamiento.

Es fácil de implementar y eficiente cuando se trabaja con un numero de datos muy pequeño.

* Aplicaciones

Al tener una complejidad n^2, no hay aplicaciones de Bubblesort más que la de su estudio.

* Pseudocodigo
* procedure bubbleSort( list : array of items )
* loop = list.count;
* for i = 0 to loop-1 do:
* swapped = false
* for j = 0 to loop-1 do:
* /\* compare the adjacent elements \*/
* if list[j] > list[j+1] then
* /\* swap them \*/
* swap( list[j], list[j+1] )
* swapped = true
* end if
* end for
* /\*if no number was swapped that means
* array is sorted now, break the loop.\*/
* if(not swapped) then
* break
* end if
* end for
* end procedure return list

1. Shellsort

* Como funciona

El algoritmo Shell sort mejora el ordenamiento por inserción comparando elementos separados por un espacio de varias posiciones. Esto permite que un elemento haga "pasos más grandes" hacia su posición esperada. Los pasos múltiples sobre los datos se hacen con tamaños de espacio cada vez más pequeños. El último paso del Shell sort es un simple ordenamiento por inserción, pero para entonces, ya está garantizado que los datos del vector están casi ordenados.

* Ventajas

Es uno de los algoritmos mas eficientes para ordenar un numero grande de datos

No requiere memoria adicional en comparación a otros algoritmos

* Aplicaciones

Mostrar los ficheros de un directorio

Mostrar un listado ordenado del contenido de una base de datos

Ordenar los resultados de búsqueda de internet

* Pseudocodigo
* procedure shellSort()
* A : array of items
* /\* calculate interval\*/
* while interval < A.length /3 do:
* interval = interval \* 3 + 1
* end while
* while interval > 0 do:
* for outer = interval; outer < A.length; outer ++ do:
* /\* select value to be inserted \*/
* valueToInsert = A[outer]
* inner = outer;
* /\*shift element towards right\*/
* while inner > interval -1 && A[inner - interval] >= valueToInsert do:
* A[inner] = A[inner - interval]
* inner = inner - interval
* end while
* /\* insert the number at hole position \*/
* A[inner] = valueToInsert
* end for
* /\* calculate interval\*/
* interval = (interval -1) /3;
* end while
* end procedure