Taller 1

Profesor German Hernandez

Algoritmos

23 de septiembre de 2018

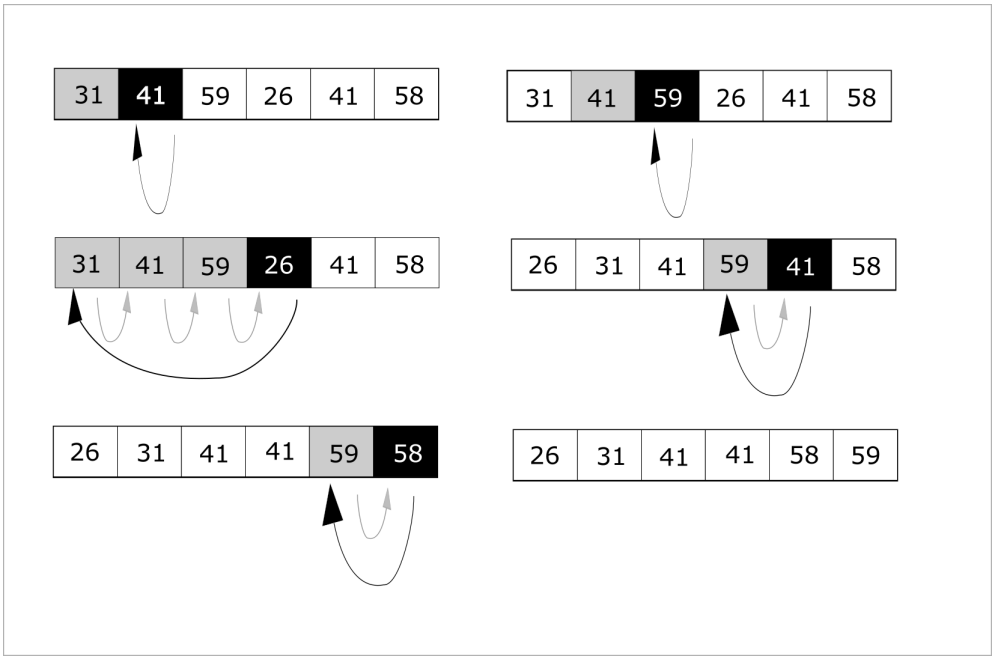
1.

1.1. 1.2.1

Motores de busqueda. En cualquier motor de busqueda se deben utilizar algoritmos de gran eficiencia ya que el tiempo que se demora la busqueda debe ser lo mas bajo posible.

2.

2.1. 2.1.1



2.2. 2.1.2

```
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. 2.1.3

```
    BUSQUEDA(A, v):
        for i = 1 to A.length
        if A[i] == v
        return i
return NIL
```

Al principio de cada iteración el subarray A[1..i-1] consiste de elementos diferentes de v.

- Inicialización: Inicialmente el subarray esta vacio.
- Mantenimiento: En cada paso, sabemos que A[1..i-1] no contiene v. Comparamos con A[i]. Sin son iguales retornamos i, lo que nos lleva a un resultado correcto. En otro caso, continuamos con el siguiente paso. Como $A[i]$ es diferente de v entonces al añadirlo al subarray $A[1..i-1]$ la invariante se mantiene.
- Terminación: La repetición termina cuando $i > A.length$. Como i incrementa en 1 sabemos que todos los elementos en A han sido revisados y que v o se encuentra entre ellos. Entonces retornamos NIL.

2.4. 2.1.4

```
SUMA_BINARIA(A, B):
  C = new integer[A.length + 1]

  carry = 0
  for i = 1 to A.length
      C[i] = (A[i] + B[i] + carry) %2  // residuo
      carry = (A[i] + B[i] + carry) / 2 // cociente
  C[i] = carry

  return C
```