4. Operaciones sobre vectores

1. Introducción
2. Búsqueda
3. Actualización de vectores ordenados y no ordenados
    1. Inserción
    2. Modificación
    3. Borrado
4. Métodos de Ordenación
    1. Cuadráticos
    2. Logarítmicos

Bibliografía:

- Clavel y Biondi.
- Weiss.
- Wirth.

4. Operaciones sobre vectores

Objetivos:

- Identificar los esquemas básicos de inserción, modificación y borrado en estructuras lineales
- Reforzar el concepto de programación modular: reutilizar operaciones ya definidas
- Conocer los esquemas básicos de ordenación interna: métodos cuadráticos y los principales esquemas de ordenación interna de complejidad logarítmica
- Distinguir cuándo deben emplearse cada uno de ellos
- Relacionar los esquemas de ordenación con los métodos de diseño de algoritmos ya vistos

1. Introducción: Recordatorio

- Conjunto de elementos del mismo tipo conceptualmente relacionados.
- Los elementos están indexados: Se accede a ellos a través de un índice.

Definiciones

- V: Vector [1..M, 1..N] de numérico;
- M: Vector [1..P, 1..Q] de carácter;

Operaciones elementales:

- **Acceso / Asignación**
  - V(1)
  - Y := V(i) + V(i-1);
  - V(1) := 1;
- **Lectura**
  - Leer V(i);
  - Leer_f (f, V(i));
- **Escritura**
  - Escribir V(i+1), x, “Cadena”;
  - Escribir_f (f, V(j));

Operaciones no elementales:

- Búsqueda
- Actualización (Inserción, Modificación y Borrado)
- Ordenación

2. Búsqueda

Hay que distinguir vectores ordenados o no.

2.1. Búsqueda en vectores no ordenados: búsqueda secuencial

```
Algoritmo Búsqueda_secuencial_1 (V, N, X, Posición, Existe) es
V: Vector [1..M] de T; {p. dato}
N: numérico; {p. dato; posiciones ocupadas}
X: T; {p. dato}
Posición: numérico; {p. resultado}
Existe: lógico; {p. resultado}
Inicio
Posición := 1;
Existe := falso;
mientras  Posición <= N y Existe = falso hacer
si V(Posición) = X entonces
    existe := Cierto;
sino
    Posición := Posición + 1;
fin si
fin mientras;
Fin
```

Búsqueda secuencial vers. 2

```
Algoritmo Búsqueda_secuencial_1_1(V, N, X, Posición, Existe) es
V: Vector [1..M] de T; {p. dato}
N: numérico; {p. dato; posiciones ocupadas}
X: T; {p. dato}
Posición: numérico; {p. resultado}
Existe: lógico; {p. resultado}
Inicio
Posición := 1;
mientras Posición< N y V(Posición) <> X hacer Posición := Posición +
    1;
fin mientras;
si V(Posicion) = X entonces
    existe := Cierto;
sino
    existe := falso;
fin si;
Fin
```

2.2. Búsqueda en vectores ordenados: búsqueda secuencial

```
Algoritmo Búsqueda_secuencial_2 (V, N, X, Posición, Existe) es
V: Vector [1..M] de T; {p. dato}
N: numérico; {p. dato}
X: T; {p. dato}
Posición: numérico; {p. resultado}
Existe: lógico; {p. resultado}
Inicio
    posición := 1;
    mientras (posición < N) y (X < V(posición)) hacer
        Posición := Posición + 1;
    fin si
    fin mientras;
    si  V(Posición) = X entonces
        existe := cierto;
    sino
        existe := falso;
    fin si;
Fin
```

2.3. Búsqueda en vectores ordenados: búsqueda binaria o dicotómica

```
Algoritmo Búsqueda_binaria_basica (V, N, X, pos) es
V: Vector [1..M] de T; {p. dato}
N: numérico; {p .dato}
X: T; {p. dato}
Pos: numérico; {p. resultado}
Inf, Sup: numérico; {var. internas}
Inicio
Inf := 1;
Sup := N;
Mientras Inf <> Sup hacer
    Med := (Inf + Sup) div 2;
    si X>V(Med) entonces {entre med+1 y sup}
          Inf := Med + 1;
    Sino {Está entre Inf y Med}
          Sup := Med;
    finsi;
Fin mientras;
si v(inf) = x entonces
    Pos := inf;
sino
    pos := -inf;
fin si
Fin
```

2.3. Búsqueda en vectores ordenados: búsqueda binaria o dicotómica

```
Algoritmo Búsqueda_binaria (V, N, X, pos) es
V: Vector [1..M] de T; {p. dato}
N: numérico; {p .dato}
X: T; {p. dato}
Pos: numérico; {p. resultado}
Inf, Sup: numérico; {var. internas}
encontrado: lógico;
Inicio
Inf := 1; Sup := N;
encontrado := falso;
Mientras Inf <> Sup AND not encontrado hacer
    Med := (Inf + Sup) div 2;
    Si X = V(med) entonces
          encontrado := cierto;
    sino
      si X>V(Med) entonces {Entre Med+1 y Sup } Inf := Med + 1;
      sino {Está entre Inf y Med} Sup := Med;
      finsi;
    finsi;
Fin mientras;
Si encontrado
    entonces  Pos := med; {o inf}
sino si v[inf] = x entonces Pos := inf;
          sino pos := -inf;
          fin si
fin si
Fin
```

3. Actualización

- Suponemos un vector de M posiciones
- Están ocupadas N de ellas (0 <= N <= M)
  - No podemos ocupar más de M
  - No podemos ocupar menos de 0
- El algoritmo ***error()*** nos permite terminar el programa, aunque no sea necesario hacerlo
- Tras cada operación de actualización (inserción, modificación o borrado) hay que asegurar que estas premisas se sigan cumpliendo.

3.1.1. Inserción en vectores desordenados

```
Algoritmo Insertar_1 (V, N, X) es
V: Vector [1..M] de T; {p. dato-resultado}
N: Numérico; {p. dato-resultado}
X: T; {p. dato}
Inicio
Si N < M entonces
        N := N +1;
        V(N) := X;
Sino
        Error ("No hay espacio disponible");
Finsi
Fin
```

3.1.2. Eliminación en vectores desordenados

```
Algoritmo Eliminar_1 (V, N, X) es
V: Vector [1..M] de T; {p. dato-resultado}
N: Numérico; {p. dato-resultado}
X: T; {p. dato}
I, Pos: numérico;
Existe: lógico;
Inicio
Si N > 0 entonces {hay elementos}
    Búsqueda_secuencial_1 (V, N, X, Pos, Existe);
    Si Existe entonces
    {opción 1, no importa el orden en que se insertaron}
    {   V(pos) := V(N);
        N:=N-1; }
    {opción 2, si importa el orden en que se insertaron}
        N := N –1;
        Para I de Pos a N hacer
                V(I) := V(I+1);
        Fin para
    Sino
        Error ("El elemento no existe");
    Fin si
Sino
    Error ("Vector vacío");
Fin si
Fin
```

3.1.3. Modificación en vectores desordenados

```
Algoritmo Modificar_1 (V, N, X, Y, ok) es
V: Vector [1..M] de T; {p. dato-resultado}
N: Numérico; {p. dato}
X, Y : T; {p. dato}
Pos: numérico;
ok: lógico;
Inicio
Si N > 0 entonces
    Búsqueda_secuencial_1 (V, N, X, Pos, Existe);
    Si Existe entonces
        V(Pos) := Y;
    Sino
        Error ("El elemento no existe");
    Fin si
Sino {no hay elementos}
    Error ("Vector vacío");
Fin si
Fin
```

3.2.1. Inserción en vectores ordenados

```
Algoritmo Insertar_2 (V, N, X, ok) es
V: Vector [1..M] de T; {p. dato-resultado}
N: Numérico; {p. dato-resultado}
X: T; {p. dato}
ok: lógico; {p. resultado}
Pos, I: numérico;
Inicio
ok := falso;
Si N < M entonces
    Búsqueda_binaria (V, N, X, Pos);
    Si Pos < 0 entonces{se debe insertar en la posición –pos ya que la posición debe ser
    positiva}
        pos := -pos;
        N := N +1;
        Para I de N a Pos + 1 paso -1 hacer
            V(I) := V(I-1);
        Fin para;
        V(Pos) := X;
        ok := cierto;
     sino {El elemento ya existía en v(pos)}
        Error ("El elemento ya existía");
    Fin si;
sino
    Error ("No hay espacio disponible");
Fin si
Fin
```

3.2.2. Eliminación en vectores ordenados

```
Algoritmo Eliminar_2 (V, N, X, ok) es
V: Vector [1..M] de T; {p. dato-resultado}
N: Numérico; {p. dato-resultado}
X: T; {p. dato}
ok: lógico; {p. resultado}
I, Pos: numérico;
Inicio
Si N > 0 entonces {hay elementos}
    Búsqueda_binaria (V, N, X, Pos);
    Si pos > 0 entonces
        N := N –1;
        Para I de Pos a N hacer
            V(I) := V(I+1);
        Fin para
    Sino
        Error (“El elemento no existe”);
    Fin si
Sino
    Error (“Vector vacío”);
Fin si
Fin
```

3.2.3.1. Modificación de vectores ordenados

```
Algoritmo Modificar_2 (V, N, X, Y, ok) es
V: Vector [1..M] de T; {p. dato-resultado}
N: Numérico; {p. dato}
X, Y: T; {p. dato}
ok: lógico;  {p. resultado}
Inicio
Eliminar_2 (V, N, X, ok); {O (n+log n)}
si ok entonces {se pudo eliminar}
    Insertar_2 (V, N, Y, ok); {O (n+log n)}
fin si
Fin
```

3.2.3.2. Modificación en vectores ordenados

```
Algoritmo Modificar_3 (V, N, X, Y, ok) es
V: Vector [1..M] de T; {p. dato-resultado}
N: Numérico; {p. dato}
X, Y: T; {p. dato}
ok: lógico; {p. resultado}
pos: numérico; {variable interna}
Inicio
ok := cierto;
Si N = 0 entonces
    ok := falso;
    Escribir "Vector Vacío";
Sino
    Búsqueda_binaria (V, N, X, Pos); {O(log n)}
    Si Pos > 0 entonces
        V(Pos) := Y;
        {¿es necesario reordenar?}
        {¿ordenar (V, N);?}
    Sino
        ok := falso;
        Escribir "El elemento no existe";
    Fin si
Fin si
Fin
```

¿ES NECESARIO REORDENAR?

En este momento sabemos que se cumple:

V(Pos) := Y;
V(i) ≤ V(i+1): 1 ≤ i < pos-1
V(i) ≤ V(i+1): pos +1 ≤ i < N

¿Qué casos se pueden dar?

a) V(pos-1) ≤ V(pos) ≤ V(pos+1) ó N = 1
b) V(pos) < V(pos-1) y N > 1
c) V(pos) > V(pos+1) y N > 1

¿Qué hay que hacer?

a) Nada. El vector sigue ordenado.
b) y c) → Es necesario reordenar la mitad afectada.

¿Se puede preguntar directamente
V(pos) < V(pos-1) ó V(pos) > V(pos+1)?

- Si N> 1 y pos = 1 sólo se puede preguntar ¿V(pos) > V(pos+1)?
- Si N> 1 y pos = N sólo se puede preguntar ¿V(pos) < V(pos-1)?
- Si N>1 y pos ≠ 1 y pos ≠ N entonces deben preguntarse ambos casos.

```
si N > 1 entonces {no hemos terminado}
     si pos = 1 entonces
          si V(pos) > V(pos+1) entonces
             {Insertar V(pos) en pos+1..N}
          fin si
     sino {pos ≠ 1}
          si pos = N entonces
               si V(pos) < V(pos - 1) entonces
                  {Insertar V(pos) en 1..pos-1}
               fin si;
          sino {pos ≠ 1 y pos ≠ N}
               si V(pos) > V(pos+1) entonces
                  {Insertar V(pos) en pos+1..N}
               sino
                    si V(pos) < V(pos - 1) entonces
                    {Insertar V(pos) en 1..pos-1}
                    fin si;
               finsi
          fin si;
     fin si; {pos = 1}
fin si
{CADA INSERCIÓN IMPLICA DESPLAZAMIENTOS: 0(n/2)}
```