Otros algoritmos. Búsqueda binaria

- Hay muchas situaciones donde es necesario hacer repetidas búsquedas de elementos dentro de una colección.
- Si la colección de datos no está ordenada, debe realizarse una búsqueda exhaustiva (es decir, comenzar por un extremo y buscar el elemento hacia el otro extremo, bien hasta que se encuentra, o en caso de que no esté, hasta que se llega al final) ⇨ Coste lineal
- Si la colección de datos está ordenada, se dispone de una estrategia muy eficiente: la búsqueda dicotómica ⇨ Coste logarítmico

Otros algoritmos. Búsqueda binaria

- Estrategia del algoritmo:

- Se considera que se busca en un intervalo del array `v`, por ejemplo desde la posición `i` hasta la `j`, `v[i..j]`. Inicialmente, `v[0..n-1]`, donde `n` representa el número de elementos del array (`v.length`).

- Consiste en fijarse en la posición central: `m=(i+j)/2`, y decidir según los tres casos posibles:
  - `x = v[m]` ⇨ Acabar, ya que se ha encontrado `x` en `v`
  - `x < v[m]` ⇨ Buscar en el subarray `v[i..m-1]`
  - `x > v[m]` ⇨ Buscar en el subarray `v[m+1..j]`

Otros algoritmos. Búsqueda binaria

- Implementación

```
  /** El array está ordenado ascendentemente */
static int encBinIter(int[] v, int x){
      int i=0, j=v.length-1, mitad=0;
      boolean encontrado=false;
      while(i<=j && !encontrado){
            mitad = (i+j)/2;
            if (x==v[mitad]) encontrado = true;
            else if (x<v[mitad]) j = mitad-1;
                 else i = mitad+1;
      }
      if (encontrado) return mitad;
      else return -1;
}
```

Otros algoritmos. Búsqueda binaria

- Costes
- La función de coste depende del tamaño del array.
- Se distinguen las siguientes instancias significativas:
  - Caso mejor: El elemento a buscar está donde se realiza la primera comparación (en la posición central). En este caso, el bucle se ejecuta sólo una vez. El coste es constante.

    $$T^m(n) \in \Theta(1) \Rightarrow T(n) \in \Omega(1)$$

  - Caso peor: El elemento no se encuentra en el array. Como cada vez se parte por la mitad el intervalo donde buscarlo, el número de pasos o veces que se repite el bucle es $\log_2(n)$.

    $$T^p(n) = \log_2(n) + 1 \in \Theta(\log n) \Rightarrow T(n) \in O(\log n)$$