Algunas Veces puede ser más facil procesar los datos si están organizados.

Ejemplos de Problemas:

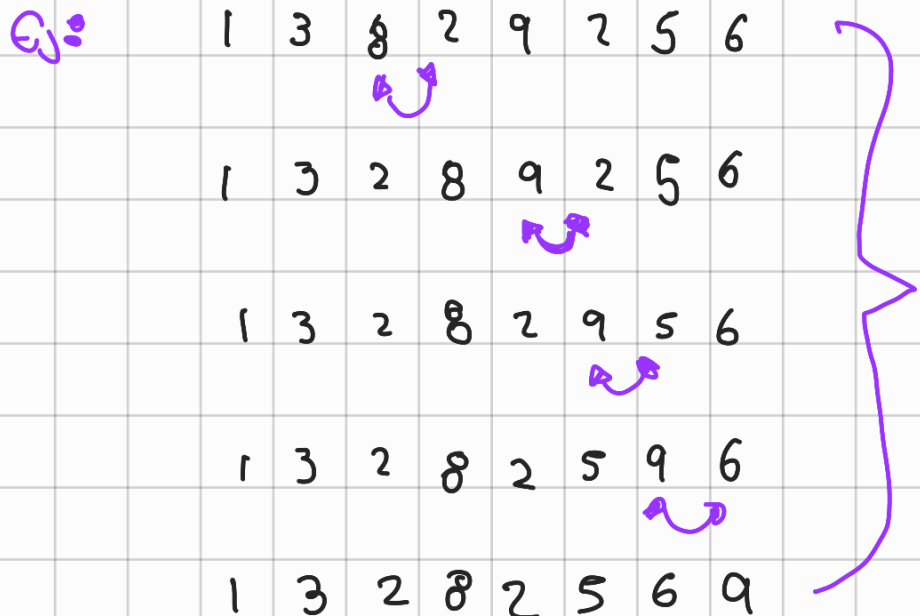
- ¿Un arreglo contiene 2 elementos iguales?
- ¿Cuál es el elemento más frecuente?

Algoritmos de Sorting $O(n^2)$

Bubble Sort → mejor de los casos: $O(n)$
Peor de los casos: $O(n^2)$

```
f(i,0,n)
  f(j,0,n-1)
      if(vec[j]>vec[j+1])
          swap(vec[j],vec[j+1]);
  }
}
```

Ej:



Un Ciclo de Bubble sort

→ Consiste en hacer n ciclos a través del arreglo, intercambiando elementos que no están en un orden consecutivo

Inversiones → Cuando los elementos no están en un orden consecutivo. Ejemplo:

↓

Nos indican que tanto costo tiene ordenar un arreglo.

1 2 2 6 3 5 9 8

Inversión 1 = (6,3)
Inversión 2 = (6,5)
Inversión 3 = (9,8)

Algoritmos de Sorting O(n log n)

Es posible sortear arreglos de una forma más eficiente si NO nos limitamos a sortear elementos de forma consecutiva.

Un ejemplo de algoritmo que hace uso de esto es el:

Merge Sort → Se basa en la recursividad.

Ordena un subarreglo [a...b] siguiendo estos pasos:

① Si a=b no hace nada porque el subarreglo ya está ordenado.

② Calcula la posición del elemento del medio $k = \lfloor (a+b)/2 \rfloor$

③ Ordena de manera recursiva el subarreglo [a...k]

④ Ordena de manera recursiva el subarreglo [k+1.....b]

⑤ Combina el subarreglo ordenado [a..k] y el subarreglo Ordenado [k+1 ... b] en un arreglo ordenado [a... b]

Es efectivo porque parte a la mitad el subarreglo por cada paso. La recursión consiste en O(log n) niveles y procesar por cada nivel toma un tiempo de O(n).

Combinar [a... k] y [k+1 ... b] toma un tiempo lineal porque los subarreglos ya están ordenados.

Ejemplo:

1 3 6 2 8 2 5 9

Se divide en:

[1, 3, 6, 2] [2, 8, 5, 9]

Luego los subarreglos se ordenan de forma recursiva

[1, 2, 3, 6] [2, 5, 8, 9]

Y el algoritmo mezcla los dos subarreglos para que al final salga este arreglo organizado

[1, 2, 3, 4, 5, 6, 7, 8, 9]

Sorkando Lower Bounds (Límites Inferiores)