# Intro a Algoritmia

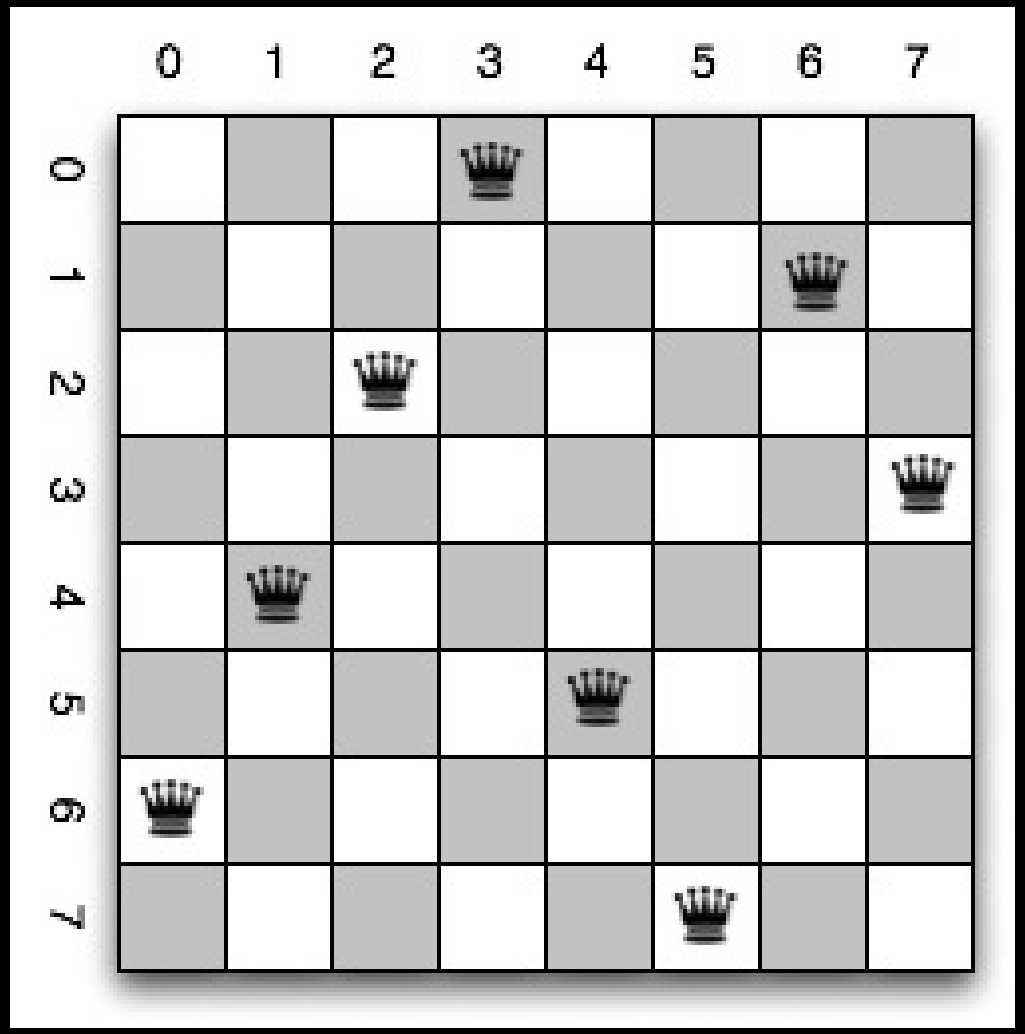
# Semana 3

# Backtracking

- Determinar algoritmos para encontrar soluciones a problemas específicos por **ensayo y error**
- **Descomponer** el proceso de ensayo y error en las **tareas parciales**
- A menudo, estas tareas se expresan naturalmente en **términos recursivos** y consisten en la **exploración** de un número finito de **sub-tareas**

# N-Queens

- `n` reinas deben ser colocadas en un tablero de ajedrez de tal manera que no se puedan atacar entre sí



# Sudoku

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

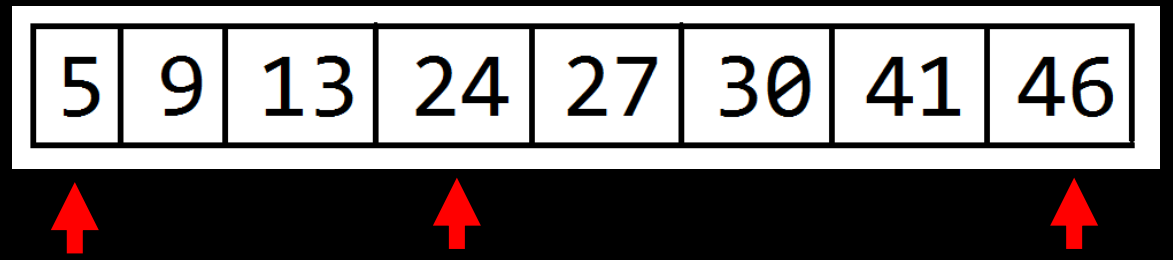
# Divide y vencerás

- Estrategia introducida por los romanos en el siglo IV antes de Cristo.
  - Divide et impera
- Como estrategia algorítmica es una de las más exitosas.
  - Dividir un problema en instancias más pequeñas
  - Resolver cada instancia de manera independiente
  - Combinar las soluciones para solucionar problema original.

# Búsqueda binaria

```
x = 41
ini = 0
fin = 7
```

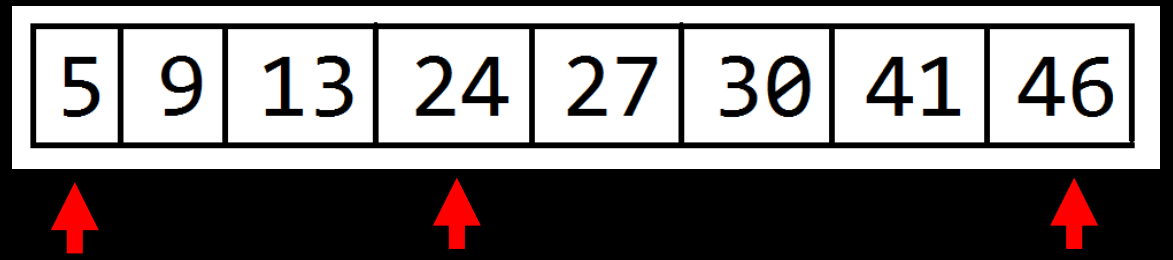
```
Mid = (ini+fin)/2
Mid = (0+7)/2
Mid = 3
```



# Búsqueda binaria

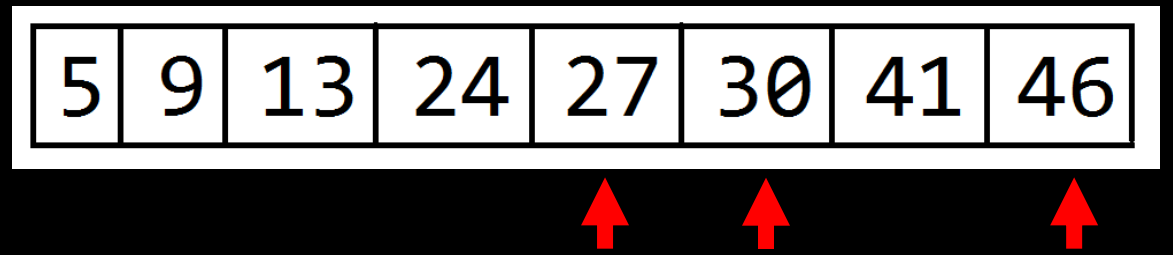
```
x = 41
ini = 0
fin = 7
```

```
Mid = (ini+fin)/2
Mid = (0+7)/2
Mid = 3
```




```
x = 41
ini = 4
fin = 7
```

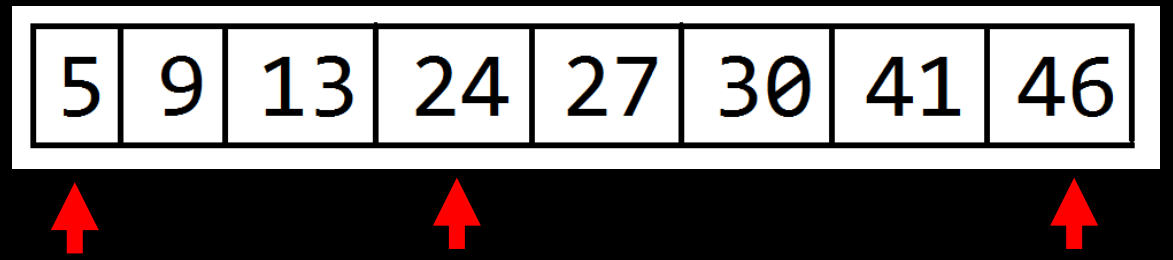
```
Mid = (ini+fin)/2
Mid = (4+7)/2
Mid = 5
```



# Búsqueda binaria

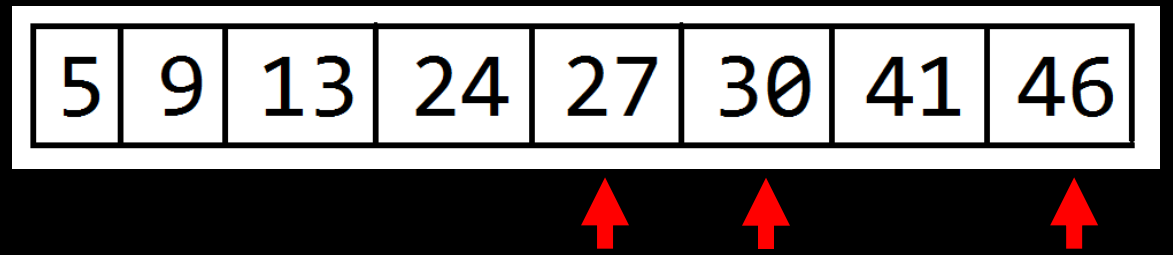
```
x = 41
ini = 0
fin = 7
```

```
Mid = (ini+fin)/2
Mid = (0+7)/2
Mid = 3
```



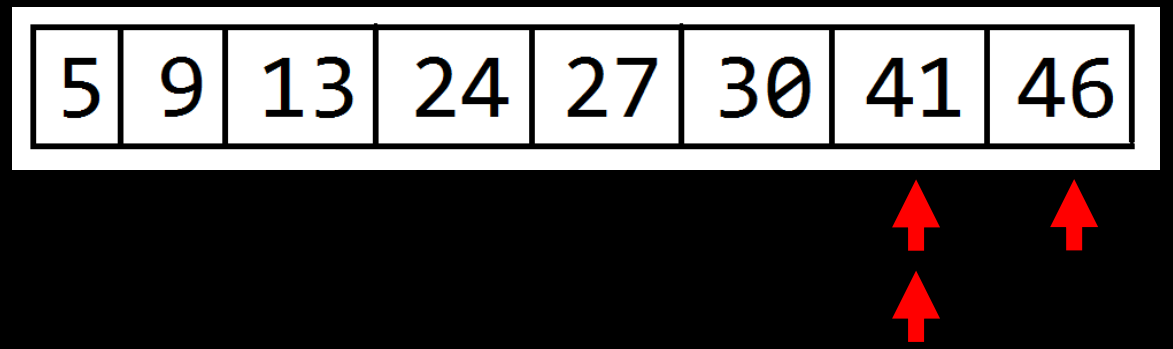

```
x = 41
ini = 4
fin = 7
```

```
Mid = (ini+fin)/2
Mid = (4+7)/2
Mid = 5
```




```
x = 41
ini = 6
fin = 7
```

```
Mid = (ini+fin)/2
Mid = (6+7)/2
Mid = 6
```



# Código

```
3  int main(){
4
5      int ini = 0, fin = n-1, mid;
6      int arreglo[10] = {5,9,12,17,20,23,27,31,34,36};
7      int x = 12, pos = -1;
8      while(ini <= fin){
9          mid = (ini+fin)/2;
10         if(arreglo[mid] == x){
11             pos = mid;
12             break;
13         }else if(x > arreglo[mid]){
14             ini = mid+1;
15         }else{
16             fin = mid-1;
17         }
18     }
19     printf("El numero se encuentra en la posicion %d\n",pos);
20     return 0;
21 }
```

# Código recursivo

```
2
3  int busquedaBinaria(int ini, int fin, int x, int arreglo[]){
4      if(ini > fin)
5          return -1;
6      int mid = (ini+fin)/2;
7      if(arreglo[mid] == x){
8          return mid;
9      }else if(x > arreglo[mid]){
10         return busquedaBinaria(mid+1,fin,x,arreglo);
11     }else{
12         return busquedaBinaria(ini,mid-1,x,arreglo);
13     }
14 }
15
```

# Ejercicio 1

2. Dado un arreglo de 0's y 1's el cual tiene todos los 1's primero seguido de todos los 0's. Encontrar el número de 0's.

   Ejemplos:

   Entrada: {1, 1, 1, 0, 0, 0, 0}

   Output: 4

   Entrada: {1, 1, 1, 1, 1}

   Salida: 0

# Ejercicio 2

5. Dado un arreglo de enteros el cual primero crece y luego decrece. Encontrar el máximo valor en el arreglo.

   Ejemplo:

   Entrada:{8, 10, 20, 80, 100, 200, 400, 500, 3, 2, 1}

   Salida: 500

# Ejercicio 3

3. Un arreglo está rotado en algún punto desconocido. Encontrar el elemento mínimo del arreglo. Asumir que todos los elementos son distintos.

   Ejemplo:

   Entrada: {5, 6, 1, 2, 3, 4}

   Salida: 1

# Otros ejemplos

- Quicksort

  https://www.youtube.com/watch?v=aQiWF4E8flQ

- Mergesort

  https://www.youtube.com/watch?v=EeQ8pwjQxTM&t=147s

# Fin de la semana 3