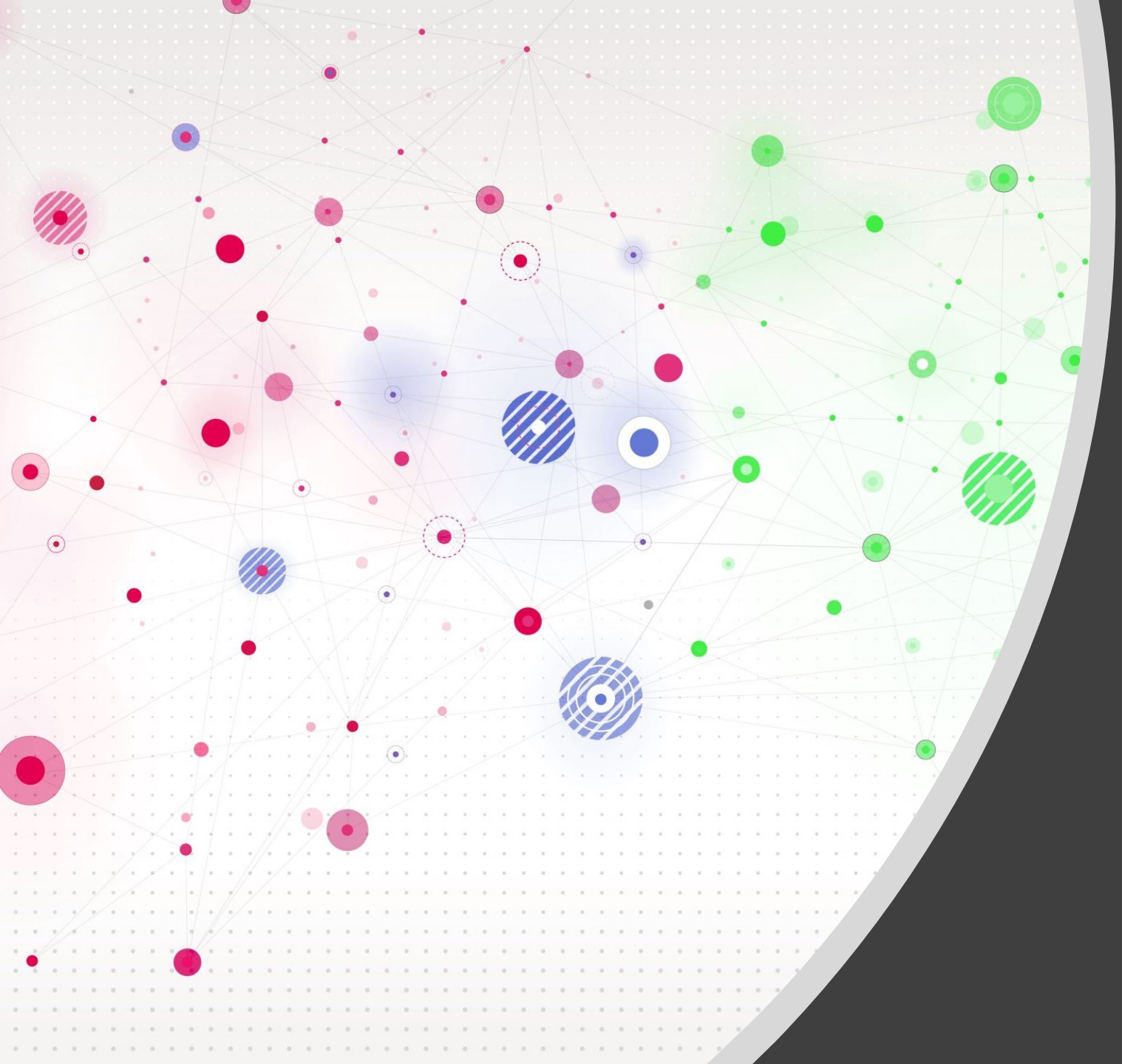

QuickSort

Rastreamento

Método:
QuickSort

Vetor desordenado:

9	8	1	4	3

quicksort (v, 0, 4)

j = separa (v, 0, 4)

```
1  // Recebe vetor v[p..r] com p <= r. Rearranja
2  // os elementos do vetor e devolve j em p..r
3  // tal que v[p..j-1] <= v[j] < v[j+1..r].
4  int separa (int v[], int p, int r) {
5     int c = v[r]; // valor do pivô
6     int t, j = p; //j guarda a posição do pivô
7  //Rearranjar o vetor em valores menores e maiores que o pivô:
8     for (int k = p; k < r; ++k)
9        if (v[k] <= c) {
10          t = v[j], v[j] = v[k], v[k] = t;
11          ++j; //Atualizar a posição do pivô
12       }
13 // Coloca o pivô (v[r]) no seu devido lugar (na posição j):
14    t = v[j], v[j] = v[r], v[r] = t;
15    return j;
16 }
```

separa (v, 0, 4)

c = 3

j = 0

k = 0 --- 3

k = 0 → F

k = 1 → F

k = 2 (1 <= 3) → V

1	8	9	4	3

j = 1

k = 3 → F

1	3	9	4	8

Retorne 1

Método:
QuickSort

Vetor desordenado:

1	3	9	4	8

quicksort (v, 0, 4)

j = separa (v, 0, 4) [valor retornado: 1]

quicksort (v, 0, 0)

quicksort (v, 0, 0)

Não entra no if. Logo não faz nada.

```
1  // Esta função rearranja qualquer vetor
2  // v[p..r] em ordem crescente.
3
4  void quicksort (int v[], int p, int r)
5  {
6     if (p < r) {
7        int j = separa (v, p, r);
8        quicksort (v, p, j-1);
9        quicksort (v, j+1, r);
10    }
11 }
```

Método:
QuickSort

Vetor desordenado:

1	3	9	4	8

quicksort (v, 0, 4)

j = separa (v, 0, 4) [valor retornado: 1]

quicksort (v, 0, 0)

quicksort (v, 2, 4)

quicksort (v, 2, 4)

j = separa (v, 2, 4)

separa (v, 2, 4)

c = 8

j = 2

k = 2 --- 3

k = 2 → F

k = 3 (4 <= 8) → V

1	3	4	9	8

j = 3

1	3	4	8	9

Retorne 3

```
1  // Recebe vetor v[p..r] com p <= r. Rearranja
2  // os elementos do vetor e devolve j em p..r
3  // tal que v[p..j-1] <= v[j] < v[j+1..r].
4  int separa (int v[], int p, int r) {
5     int c = v[r]; // valor do pivô
6     int t, j = p; //j guarda a posição do pivô
7  //Rearranjar o vetor em valores menores e maiores que o pivô:
8     for (int k = p; k < r; ++k)
9        if (v[k] <= c) {
10          t = v[j], v[j] = v[k], v[k] = t;
11          ++j; //Atualizar a posição do pivô
12       }
13 // Coloca o pivô (v[r]) no seu devido lugar (na posição j):
14    t = v[j], v[j] = v[r], v[r] = t;
15    return j;
16 }
```

Método:
QuickSort

Vetor ORDENADO:

1	3	4	8	9

quicksort (v, 0, 4)

j = separa (v, 0, 4) [valor retornado: 1]

quicksort (v, 0, 0)

quicksort (v, 2, 4)

quicksort (v, 2, 4)

j = separa (v, 2, 4) [valor retornado: 3]

quicksort (v, 2, 2)

quicksort (v, 2, 2)

Não entra no if. Logo não faz nada.

```
1  // Esta função rearranja qualquer vetor
2  // v[p..r] em ordem crescente.
3
4  void quicksort (int v[], int p, int r)
5  {
6     if (p < r) {
7        int j = separa (v, p, r);
8        quicksort (v, p, j-1);
9        quicksort (v, j+1, r);
10    }
11 }
```

Método:
QuickSort

Vetor ORDENADO:

1	3	4	8	9

quicksort (v, 0, 4)

j = separa (v, 0, 4) [valor retornado: 1]

quicksort (v, 0, 0)

quicksort (v, 2, 4)

quicksort (v, 2, 4)

j = separa (v, 2, 4) [valor retornado: 3]

quicksort (v, 2, 2)

quicksort (v, 4, 4)

quicksort (v, 4, 4)

Não entra no if. Logo não faz nada.

```
1  // Esta função rearranja qualquer vetor
2  // v[p..r] em ordem crescente.
3
4  void quicksort (int v[], int p, int r)
5  {
6     if (p < r) {
7        int j = separa (v, p, r);
8        quicksort (v, p, j-1);
9        quicksort (v, j+1, r);
10    }
11 }
```

Método:
QuickSort

Vetor ORDENADO:

1	3	4	8	9