PROGRAMAÇÃO E ALGORITMOS (LEII)

Universidade da Beira Interior, Departamento de Informática
Hugo Pedro Proença, 2016/2017

Resumo

- Quicksort
  - Análise
  - Complexidade
    - Pior Caso
    - Caso Médio
  -

Quicksort

- É um dos algoritmos de ordenação mais conhecidos e utilizados, devido ao seu desempenho.
  - Foi proposto em 1960, (Charles Hoare)
- È composto por duas fases principais
  - Partição
  - Ordenação
- Segue uma estratégia bastante comum no domínio da ciência computacional: “dividir para reinar”.

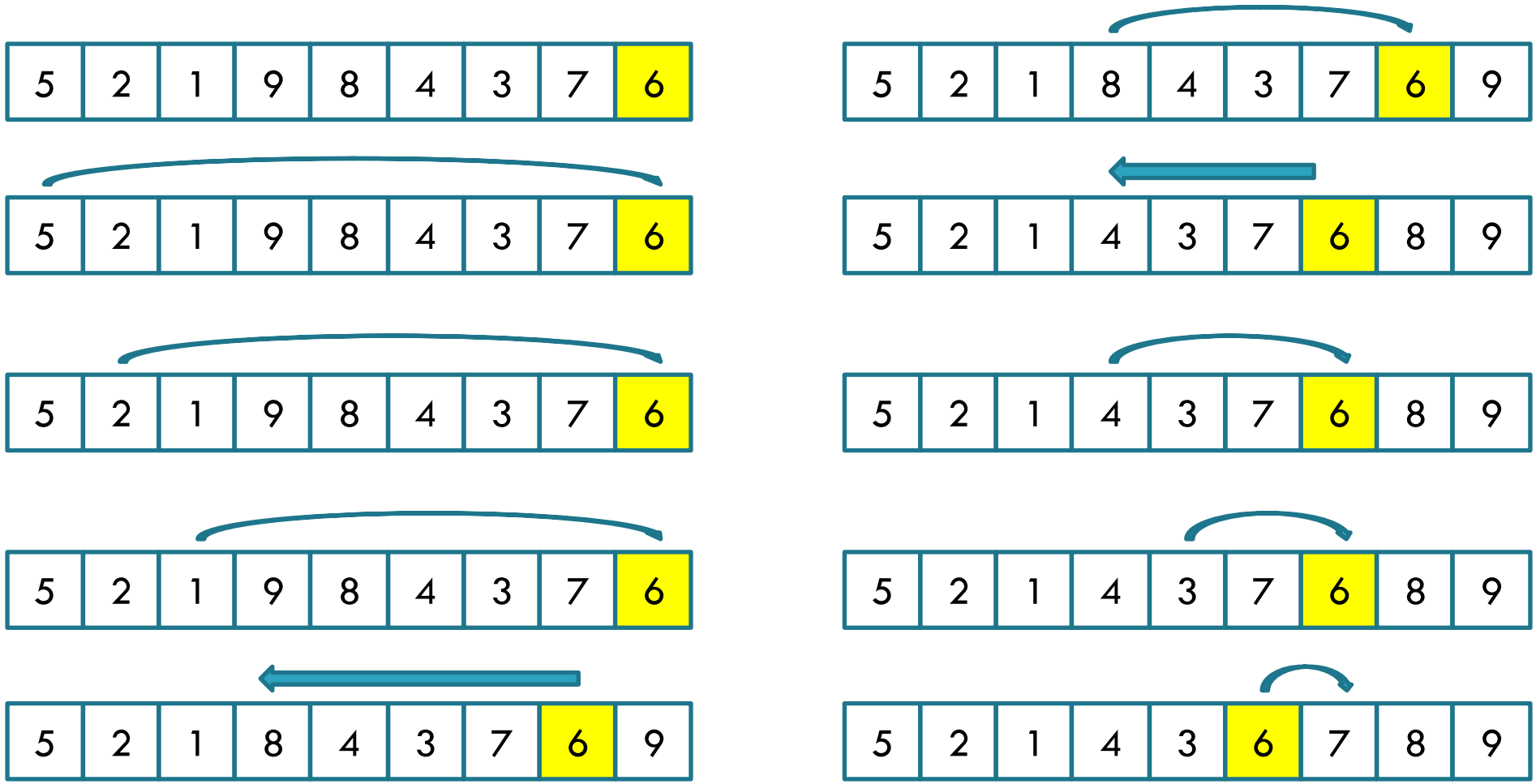
Quicksort: Algoritmo

- Escolher um elemento do conjunto (pivot)
- Ordenar o conjunto, por forma a que todos os elementos menores que o pivot estejam à sua esquerda e todo os elementos maiores à sua direita
  - Após este passo, o pivot está na sua posição natural
  - Esta é chamada a fase de partição, uma vez que agora faltará ordenar os elementos à esquerda e direita
- Aplicar o princípio da recursividade para ordenar os 2 subconjuntos menores.
  - Caso-base será quando os sub-conjuntos tiverem um único elemento.

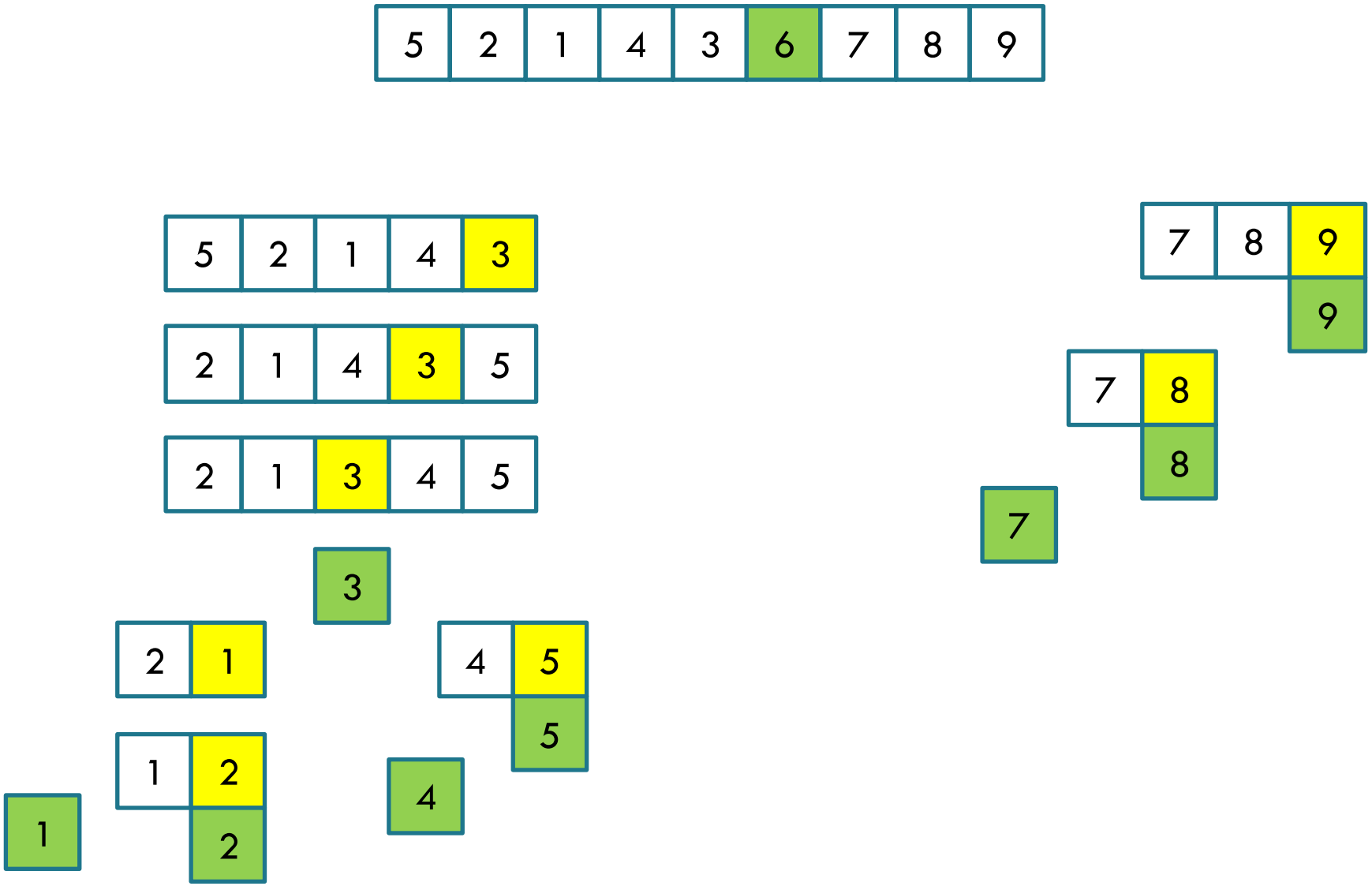
Quicksort

- O processo é sensível à escolha do elemento pivot.
- Existem variantes do algoritmo, uma vez que na sua versão mais simples está longe de ser um algoritmo "in-place".
  - Exige a alocação de "n" posições de memória, para ordenar um vector de "n" posições.
  - A alocação tem que ser encarada como uma chamada ao sistema operativo e, por conseguinte, com custo computacional elevado.

Quicksort: exemplo



Quicksort: exemplo



Quicksort: Pseudo-código

- **function** quicksort(array)
  - **var** *list* less, greater
  - **if** length(array) ≤ 1
    - **return** array
  - select and remove a pivot value *pivot* from array
  - **for each** x **in** array
    - **if** x ≤ pivot **then**
      - append x to less
    - **else**
      - append x to greater
  - **return** concatenate(quicksort(less), pivot, quicksort(greater))

Quicksort: Análise Computacional

- Qual o melhor caso do algoritmo?
  - Quando o pivot é o elemento central, que divide o problema em duas sub-partes iguais.

- Neste caso, prova-se que o algoritmo tem complexidade computacional O(n log(n))
  - "n" relativo à parte da comparação do pivot com os restantes elementos
  - Log(n) relativo à decomposição binária em sub-problemas
- No pior caso, o algoritmo terá complexidade $O(n^2)$

Quicksort: Análise Computacional

- Exercício: Instancie dois vectores de 10 elementos:
  - Um que corresponda ao melhor caso do algoritm
  - Outro que corresponda ao seu pior caso.