Introdução à Ciência da Computação II

Quick-Sort Pt. I: Algoritmo Básico & Desempenho

Prof. Ricardo J. G. B. Campello

1

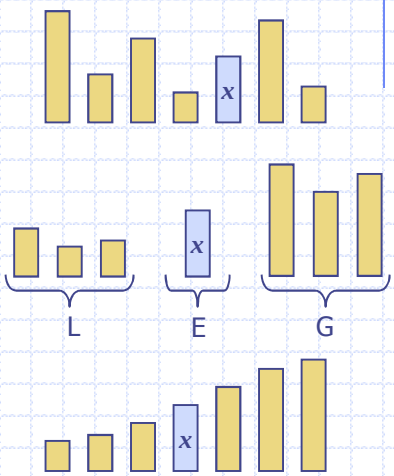
Aula de Hoje

- Quick-Sort Conceitual
- Análise de Desempenho
  - Pior Caso
  - Melhor Caso e Caso Médio
- Implementação em C (Ingênua)

2

Algoritmo Quick-Sort

- Quick-sort é um algoritmo de ordenação baseado no paradigma de **divisão e conquista**:
  - Divisão: toma aleatoriamente um elemento x (pivô) e particiona a seqüência em:
    - L : elementos menores que x
    - E : elementos iguais a x
    - G : elementos maiores que x
  - Recursão: ordena L e G
  - Conquista: junta L, E e G



3

Etapa de Divisão

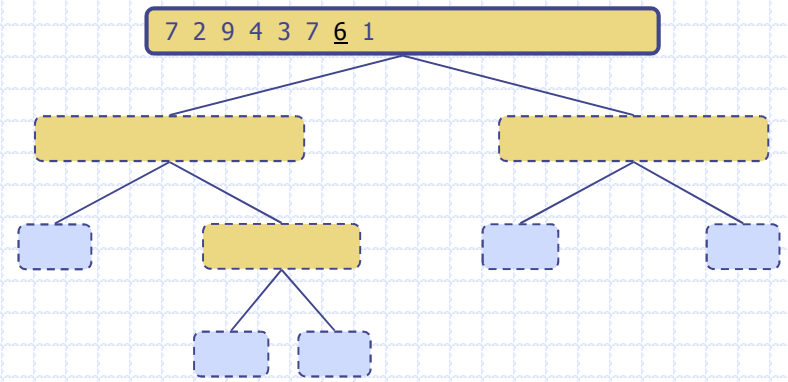
- Dado o pivô x, divide-se a seqüência de valores a serem ordenados (vetor ou lista) como segue:
  - Toma-se cada elemento y
  - Insere-se y em L, E ou G
    - Em função do resultado da comparação de y com o pivô x
- O tempo de execução é linear no tamanho n_v do vetor, ou seja, $O(n_v)$

Algoritmo *Partição*(V, n_v, p)
  Entradas: Vetor V, tamanho n_v deste vetor e posição p do pivô
  Saída: Vetores L, E e G
  L, E, G ← Novos vetores
  $cont\_l$, $cont\_g$, $cont\_e$ ← 0
  $x \leftarrow V[p]$
  para $i \leftarrow 0$ **até** $n_v - 1$ **faça**
    $y \leftarrow V[i]$
    se $y < x$
      $L[cont\_l{+}{+}] = y$
    senão **se** $y > x$
      $G[cont\_g{+}{+}] = y$
    senão
      $E[cont\_e{+}{+}] = y$
  retorne L, E, G

4

Exemplo de Execução Quick-Sort

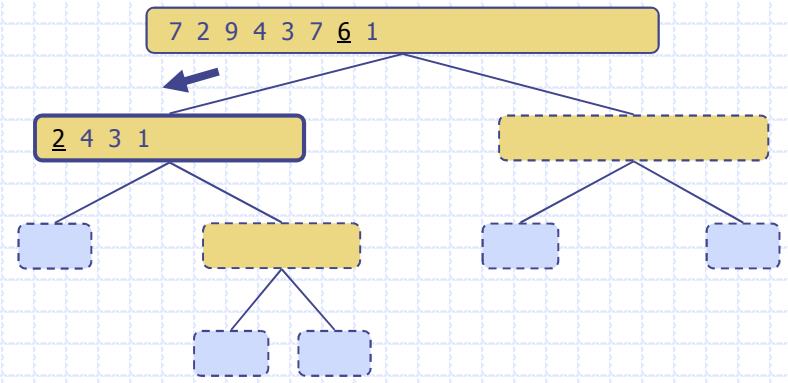
- Seleção do pivô:



5

Exemplo de Execução (cont.)

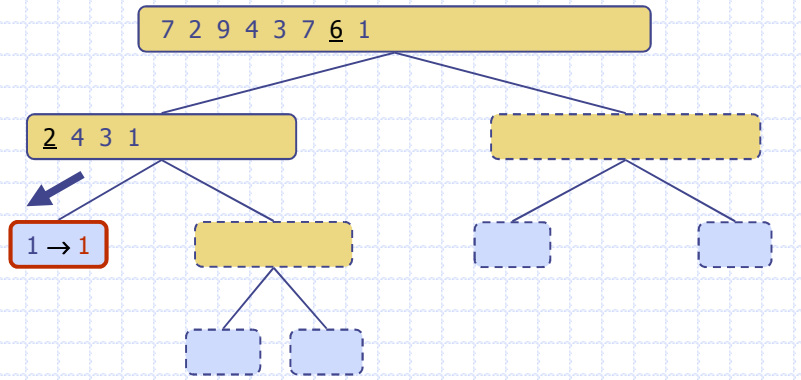
- Partição, chamada recursiva em L e seleção do pivô:



6

Exemplo de Execução (cont.)

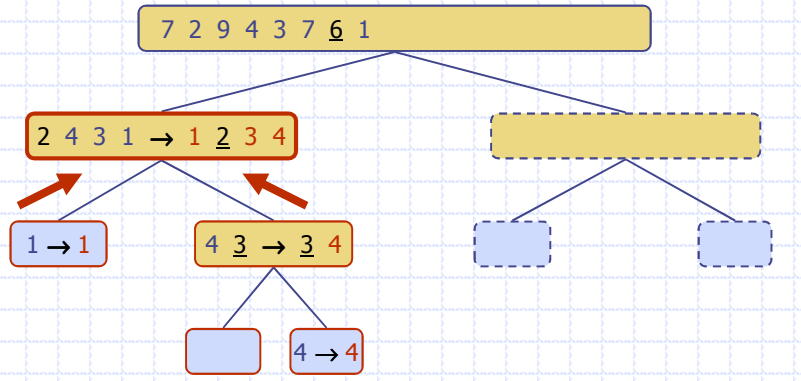
- Partição, chamada recursiva em L e condição de parada:



7

Exemplo de Execução (cont.)

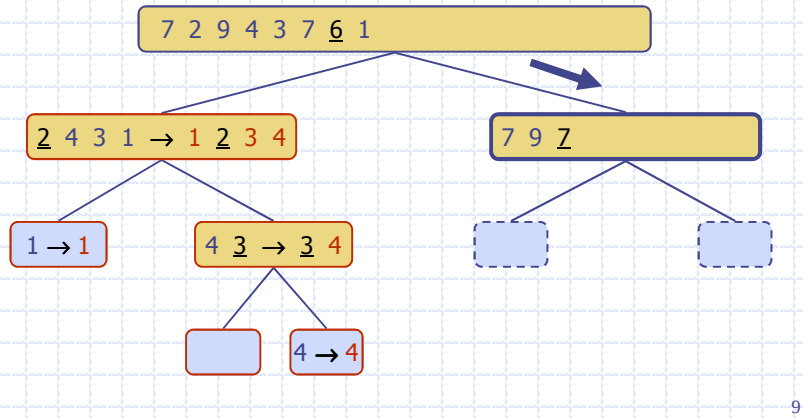
- Chamada recursiva em G, seleção do pivô, partição, chamada recursiva em G (L vazia), condição de parada, junção, junção:



8

Exemplo de Execução (cont.)

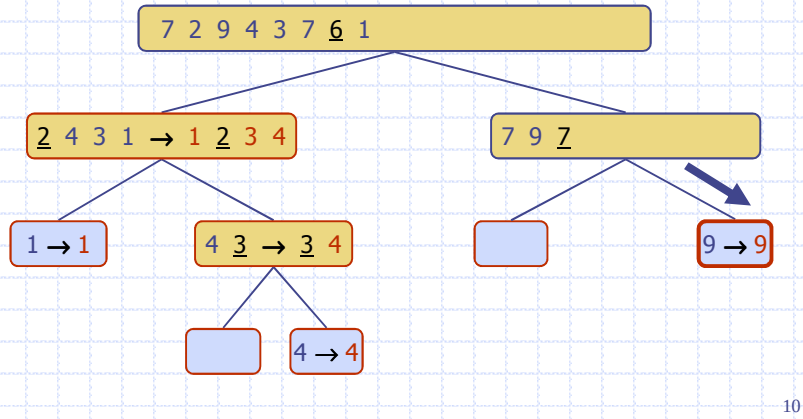
- ....



9

Exemplo de Execução (cont.)

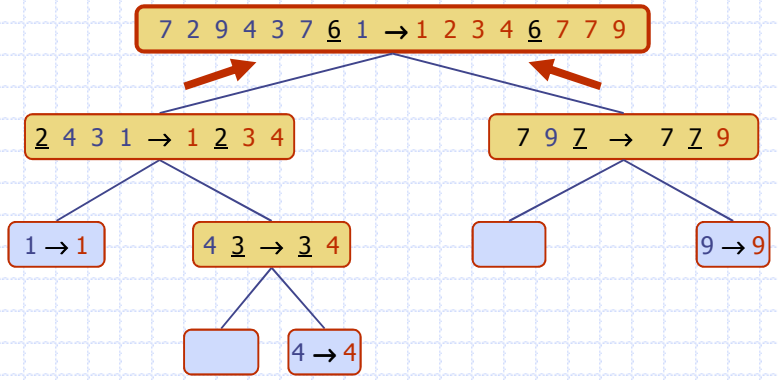
- ....



10

Exemplo de Execução (cont.)

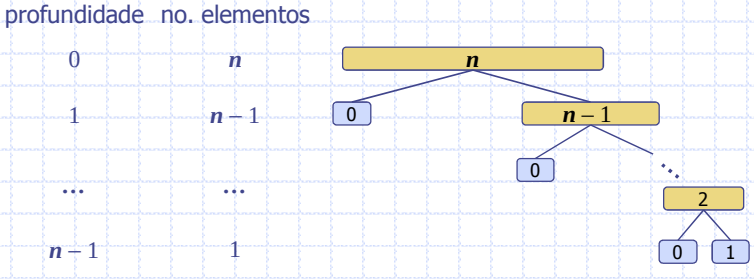
- ....



11

Tempo de Execução do Pior Caso

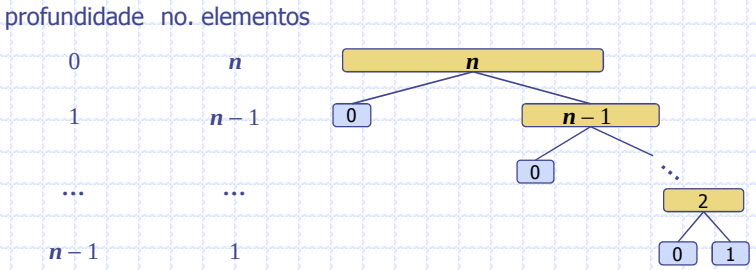
- O pior caso para quick-sort ocorre quando o pivô é sempre o menor (ou maior) elemento da seqüência, sendo este único
- Nesse caso, E possui apenas o pivô, L (ou G) é vazia e G (ou L) possui todos os demais elementos



Tempo de Execução do Pior Caso

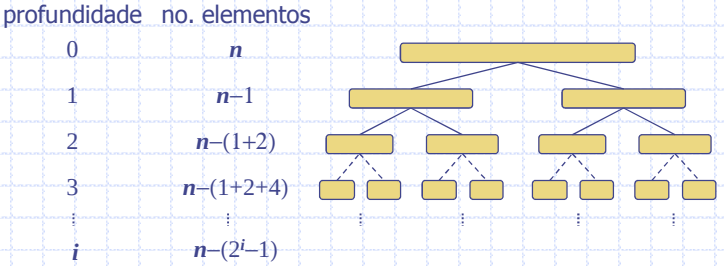
- Em cada nível i da árvore de recursão (binária), os n_i elementos naquele nível são subdivididos em L, E ou G, o que é feito em tempo linear (vide algoritmo ***Partição***)
- Logo, no pior caso o algoritmo roda em tempo:

$$O(n + n-1 + \ldots + 2 + 1) \Rightarrow O(n(n+1)/2) \Rightarrow O(n^2)$$



Tempo de Execução do Melhor Caso

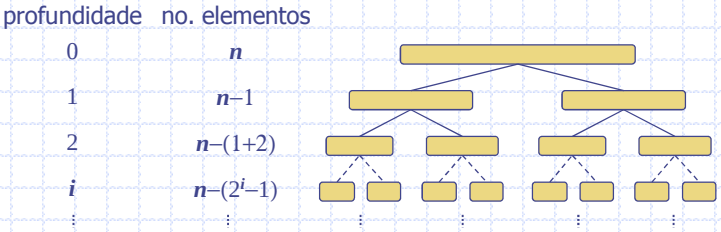
- O melhor caso para quick-sort ocorre quando o pivô é sempre tal que a seqüência seja dividida em L e G de tamanhos ≈ iguais
- Nesse caso, é simples demonstrar que a altura da árvore de recursão é $O(\log n)$



14

Tempo de Execução do Melhor Caso

- O tempo de execução em cada nível i da árvore de recursão não é maior que n, isto é, é $O(n)$
- Logo, quick-sort é $O(n \log n)$ no melhor caso
  - isto é, $\Omega(n \log n)$



15

Tempo Esperado (Caso Médio)

- Infelizmente, não podemos selecionar o pivô ideal sem pagar um preço computacional
- Felizmente, uma seleção puramente aleatória garante tempo esperado também proporcional a $n \log n$, isto é, $O(n \log n)$ em média

16

Implementação em C

- No quadro...

17

Resumo

- Algoritmo **pode ser implementado In-Place***
  - Mas às custas da **perda de Estabilidade**
  - Vide próxima aula...
- Algoritmo é eficiente, mas não garantidamente:
  - $O(n^2)$ no pior caso
    - pivô separa L e G de forma totalmente desbalanceada
  - $O(n \log n)$ no melhor caso
    - pivô separa L e G de forma homogênea
  - $O(n \log n)$ na média
- Na prática, um dos mais utilizados algoritmos de ordenação (quando estabilidade não é requerida)

18

Biblioteca em C

- **stdlib.h**
  - funções para ordenação, busca, ...
  - Por exemplo, **qsort** :

```
void  qsort( v, n, sizeof(tipo_v), comparador);
```

vetor de elementos

no. de elementos

tipo dos elementos do vetor v

nome de função que compara elementos de v (tipo_v)

19

Próxima Aula...

- Quick-Sort com Partição In-Place
- Quick-Select

20

Exercícios

- Elabore um exemplo que ilustre uma sucessão de escolhas aleatórias de pivô que levam ao pior caso de tempo de execução quick-sort
  - ilustre a execução do algoritmo através da árvore de recursão correspondente (conforme exemplo nos slides)
- Repita o exercício anterior para o melhor caso
- Mostre que a altura da árvore de recursão quick-sort é, no melhor caso, de ordem $O(\log_2 n)$

21

Bibliografia

- M. T. Goodrich & R. Tamassia, *Data Structures and Algorithms in C++/Java*, John Wiley & Sons, 2002/2005
- N. Ziviani, *Projeto de Algoritmos*, Thomson, 2a. Edição, 2004
- A. M. Tenembaum et al., *Data Structures Using C*, Prentice-Hall, 1990

22