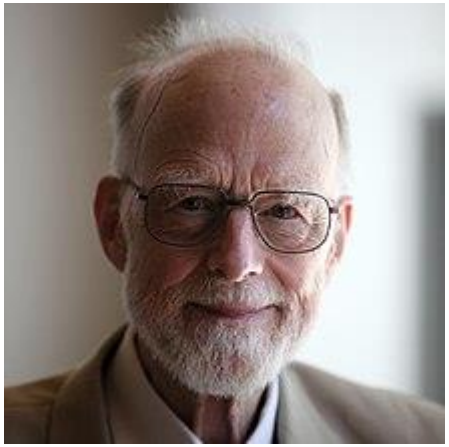
QuickSort: ordenação por partição

Fabio Lubacheski
fabio.lubacheski@mackenzie.br

QuickSort

- O algoritmo *Quicksort*, inventado por C.A.R. Hoare em 1962, na média é muito rápido em algumas raras instâncias (pior caso), o *Quicksort* pode ser tão lento quanto os algoritmos elementares (bolha, inserção e seleção).
- Hoare criou o *QuickSort* pra tentar traduzir um dicionário de inglês para russo, ordenando as palavras, tendo como objetivo reduzir o problema original em subproblemas que possam ser resolvidos mais fácil e rápido.
- Simulação de algoritmos de ordenação:

  http://nicholasandre.com.br/sorting/



QuickSort

- O pior caso do *QuickSort* ocorre para as entradas que correspondem às listas ordenadas, crescentes ou decrescentes, onde sistematicamente, após cada partição resulta uma sublista vazia.

- O coração do *Quicksort* está no subproblema da separação (=partição), que será formulada de maneira vaga: rearranjar `v[p]..v[r]` "*de modo que todos os elementos menores fiquem na parte esquerda do vetor e todos os elementos maiores fiquem na parte direita*".

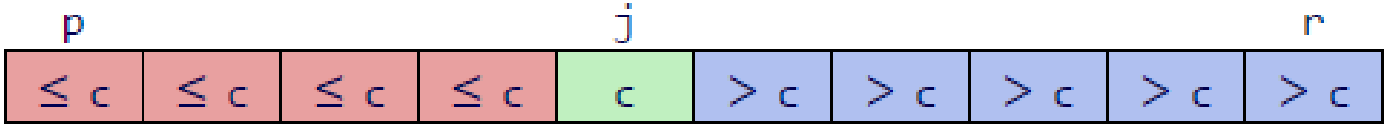
O subalgoritmo da Partição

- O ponto de partida para a solução do subproblema é a escolha de um “pivô" `c`:

- Supondo `p < r`, a função rearranja o vetor `v[p..r]` de tal que de modo que:

  `v[p..j-1]<=v[j]<v[j+1..r]`, para algum `j` em `p..r` , onde em `v[j]` estaria o pivô.

  Seria melhor se `j` ficasse a meio caminho entre `p` e `r`, mas pode ocorrer casos extremos `j=p` e `j=r`.



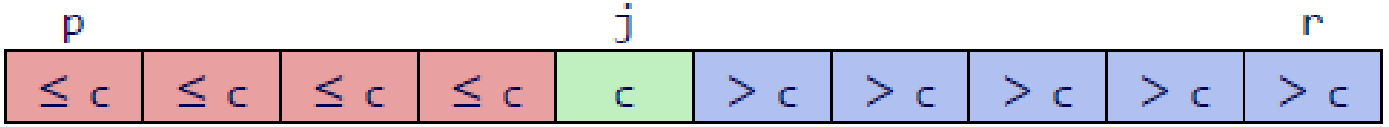
O subalgoritmo da Partição

- A ideia do algoritmo partição é varrer o vetor do início para o fim e parar o índice `i` no primeiro elemento maior encontrado em relação ao pivô. Em seguida varrer a lista do fim para o início usando o índice `j` até encontrar o primeiro elemento menor (ou igual) que o pivô. Após a parada dos dois índices, trocar os elementos de posição e continuar até que todos os elementos da lista tenham sido examinados. Ao final é retornado o índice do pivô (`j`).

- **Agora vamos implementar o Partição**

QuickSort

- Agora que resolvemos o problema da partição, podemos cuidar do QuickSort propriamente dito. O algoritmo usa a estratégia da divisão e conquista, o QuickSort recebe como parâmetro `p`, `r` e o vetor e executa recursivamente enquanto `p < r`.
- Primeiramente o algoritmo QuickSort faz a partição do vetor em seguida o QuickSort faz uma chamada recursiva para parte a esquerda do pivô e a para parte direita do pivô.



- **Agora vamos implementar o QuickSort....**

Exercícios

1) Execute o algoritmo `Partição` com a entrada

   `v=[11, 10, 4, 3, 3, 2]`

2) Faça o diagrama de execução para o QuickSort com

   `v=[5, 9, 6, 4, 3]`

3) Por que o pior caso do `QuickSort` é quando ele está ordenado em ordem crescente ou descrescente ?

4) Escreva uma versão iterativa para o método de ordenação `QuickSort`.

Fim