Lista 12 - MAC0122 Princípios de Desenvolvimento de Algoritmos - POLI

Prof. Ronaldo Fumio Hashimoto

1 Exercícios

1. (Adaptado de [1]) Quais dos vetores abaixo são *heaps*?
   - `[ 161, 41, 101, 141, 71, 91, 31, 21, 81, 17, 16 ]`
   - `[ 98, 97, 88, 81, 62, 72, 31, 70, 61, 15 ]`
2. (Extraído de [1]) Escreva uma função que determina se um vetor $v[1..m]$ é ou não um *heap*.
3. (Extraído de [2]) Considere cada letra uma inserção e cada asterisco (`*`) uma remoção do elemento com prioridade máxima em uma fila de prioridade por ordem lexicográfica. Dê a sequência de elementos retornados pela operação de remoção na fila da seguinte sequência:
   `P R I O * R * * I * T * Y * * * Q U E * * * U * E`.
4. Ilustre a execução do algoritmo *quicksort*, exibindo as operações executadas na função *partition* dada em aula para cada sub-vetor utilizando o seguinte vetor:
   - `[ 2, 4, 9, 13, 9, 3, 11, 14, 10, 1 ]`
5. Escreva uma versão do *quicksort* que ordena, por ordem lexicográfica, um vetor que armazena uma *string* (`char *`) em uma cada posição.
6. Escreva um exemplo de entrada para o *quicksort* (implementado com mostrado em aula) que gera uma ordenação não estável[1].
7. (Extraído de [1]) Escreva um algoritmo para ordenar um vetor com n `bits` (vetor em que cada elemento tem valor 0 ou 1) que gasta tempo linear em relação ao tamanho do vetor ($O(n)$).

[1] Um algoritmo de ordenação estável não altera a ordem relativa de elementos empatados, i.e., se dois elementos x e y têm o mesmo valor, mas x vinha antes de y (antes da ordenação), então x ainda virá antes de y depois da ordenação. Um exemplo da utilidade disso é mostrado no Exercício 9.

8. (Extraído de [3]) Escreva uma função que rearranja um vetor $v[p..r]$ de inteiros de modo que tenhamos $v[p..j-1] \leq 0$ e $v[j..r] > 0$ para algum j em $p..r$. Procure fazer uma função rápida que não use vetor auxiliar.

9. Ordenação estável é útil para organizar dados que contém mais de um campo. Por exemplo, ordenar alunos por notas, mas dentre os alunos de mesma nota, ordená-los em ordem alfabética.

   Um outro exemplo seria um baralho de cartas: um baralho, quando comprado novo, é organizado obedecendo a seguinte ordem (desconsiderando os coringas):

   (A♠, 2♠, ..., 10♠, J♠ Q♠, K♠, A♢, ..., K♢, A♣, ..., K♣, A♡, ..., K♡).

   Em outras palavras, um baralho é ordenado obedecendo dois parâmetros, o valor e o naipe de cada carta. Assim, dado um baralho embaralhado, podemos retorná-lo ao estado original realizando duas ordenações consecutivas: uma primeiro por valor e uma segunda (estável) por naipe.

   A sequência abaixo mostra um exemplo reduzido para um baralho com valores de A a 3:

   (a) Baralho embaralhado:
   (A♣, 3♢, A♢, A♠, 2♡, 3♠, 2♣, A♡, 2♢, 3♣, 3♡, 2♠)

   (b) Ordenação por valor (A < 2 < 3) aplicado a (a):
   (A♣, A♢, A♠, A♡, 2♡, 2♣, 2♢, 2♠, 3♢, 3♠, 3♣, 3♡)

   (c) Ordenação estável por naipe (♠ < ♢ < ♣ < ♡) aplicado a (b):
   (A♠, 2♠, 3♠, A♢, 2♢, 3♢, A♣, 2♣, 3♣, A♡, 2♡, 3♡)

   Ao contrário de uma ordenação estável, uma ordenação não estável não garante que o resultado final vai estar ordenado por ambos os parâmetros, pois a segunda ordenação pode desfazer a ordem da primeira. Por exemplo, a sequência abaixo é obtida usando um algoritmo não estável:

   - Ordenação não estável por naipe aplicado a (b):
   (A♠, 2♠, 3♠, 2♢, A♢, 3♢, 2♣, 3♣, A♣, 2♡, 3♡, A♡)

   Neste caso, 2♢ e 1♢ ficaram invertidos na ordenação não estável porque, em termos de naipe, eles estão empatados e o algoritmo não faz distinção de quem deveria vir antes.

   Exercício: dado um baralho embaralhado, implemente os seguintes algoritmos de ordenação:

   - *Selection sort*
   - *Merge sort*
   - *Quicksort*

Execute as ordenações por valor e naipe e observe o estado final do baralho. Baseado nestas observações, verifique quais dos algoritmos implementados são estáveis e quais não são.

Você pode usar o código disponível no PACA que implementa um baralho e o embaralha (basta criar os algoritmos de ordenação e chamar os algoritmos no cliente, na linha especificada com `TODO`. A função de troca (*swap*), usada no *selection sort* e no *quicksort*, já está implementada no arquivo `baralho.c`).

Referências

[1] P. Feofiloff, "Quicksort." `https://www.ime.usp.br/~pf/algoritmos/aulas/quick.html`. [Online; acessado dia 23 de junho de 2017].

[2] R. Sedgewick, "Algorithms in c—third edition," p. 309, 1998.

[3] C. G. Fernandes. `https://www.ime.usp.br/~cris/aulas/16_2_5711/listas/`. [Online; acessado dia 23 de junho de 2017].