Lista de Exercícios 2 — Filas de Prioridade

QXD0115 – Estrutura de Dados Avançada – Turma 02A – 2024.1

Prof. Atílio Gomes

2 de março de 2024

Aluno: [ ] Matrícula: [ ]

1. Uma letra significa **inserir** e um asterisco significa **remover o máximo** na sequência

   P R I O * R * * I * T * Y * * * Q U E * * * U * E.

   Dê a sequência de valores retornados pelas operações de remover o máximo.

2. Explique como usar o TAD Fila de Prioridade para implementar o TAD Pilha.

3. Explique como usar o TAD Fila de Prioridade para implementar o TAD Fila.

Heap Binário

4. Quais são os números mínimo e máximo de nós que um heap binário de altura h pode ter?

5. Mostre que um heap binário com n nós tem altura $\lfloor \log_2 n \rfloor + 1$.

6. Mostre que, em qualquer subárvore de um heap binário máximo, a raiz da subárvore contém o maior valor que ocorre em qualquer lugar nessa subárvore.

7. Em que lugar de um heap binário máximo o menor elemento poderia residir, considerando que todos os elementos sejam distintos?

8. Um vetor que está em sequência ordenada é um heap binário mínimo? Por quê?

9. A sequência $(23, 17, 14, 6, 13, 10, 1, 5, 7, 12)$ é um heap binário máximo?

10. Mostre que, em um heap binário A com n elementos, as folhas são os nós com índices $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n$.

11. Ilustre a operação `maxFixDown(A,3)` sobre o vetor $A = [27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$.

12. Com base no procedimento `maxFixDown`, escreva o pseudocódigo para o procedimento `minFixDown`, que executa a manipulação correspondente sobre um heap binário mínimo. Compare o tempo de execução de `minFixDown` com o de `maxFixDown`.

13. Qual é o efeito de chamar `maxFixDown(A,i)` quando o elemento A[i] é maior que seus filhos?

14. Qual é o efeito de chamar `maxFixDown(A,i)` para $i > A.heap\_size/2$?

15. Ilustre a operação `buildMaxHeap` no vetor $A = [5, 3, 17, 10, 84, 19, 6, 22, 9]$.

16. Mostre que existem, no máximo, $\lceil n/2^h \rceil$ nós de altura h em qualquer heap com n nós.

17. Ilustre a operação `extractMax` sobre o heap binário $A = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]$.

18. Ilustre a operação `maxHeapInsert(A,10)` sobre o heap binário $A = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]$.

19. Escreva pseudocódigos para os procedimentos `getMin`, `extractMin`, `decreaseKey` e `minHeapInsert` que implementam uma fila de prioridade mínima com um heap binário mínimo.

20. A operação `maxHeapDelete(A,i)` elimina o item no nó i do heap A. Forneça uma implementação de `maxHeapDelete(A,i)` que seja executada no tempo $O(\lg n)$ para um heap binário máximo com n elementos.

21. Podemos construir um heap chamando repetidamente `maxHeapInsert` para inserir os elementos no heap. Considere a seguinte variação do procedimento `buildMaxHeap`:

Algorithm 1 Procedimento que constrói um heap binário

```
1: procedure BUILDMAXHEAP*(A)                    ▷ A é um vetor
2:     A.heap-size = 1
3:     for i = 2 to A.length do
4:         MAXHEAPINSERT(A, A[i])
5:     end for
6: end procedure
```

(a) Os procedimentos `buildMaxHeap` e `buildMaxHeap*` sempre criam o mesmo heap quando são executados sobre o mesmo vetor de entrada? Prove que isso ocorre ou, então, dê um contraexemplo.

(b) Mostre que, no pior caso, `buildMaxHeap*` requer o tempo $O(n \lg n)$ para construir um heap binário de n elementos.

HeapSort

22. Ilustre a execução do `HeapSort` no array $A = [5, 13, 2, 25, 7, 17, 20, 8, 4]$

23. Um algoritmo de ordenação é **estável** quando números com o mesmo valor aparecem no arranjo de saída na mesma ordem em que se encontram no arranjo de entrada. Essa propriedade é importante quando os dados satélites que acompanham os elementos sendo ordenados devem ser transportados juntamente com o elemento.
Mostre que o `HeapSort` não é estável.

24. Determine empiricamente a porcentagem de tempo que o `HeapSort` gasta na etapa de construção para $N = 10^3, 10^4, 10^5, 10^6$.

25. Qual é o tempo de execução do `HeapSort` em um array A de tamanho n que já está completamente ordenado? E se ele estiver em ordem decrescente?

26. Mostre que o tempo de execução do `HeapSort` no pior caso é $\Omega(n \lg n)$.