Atividade 03 - gabarito

sexta-feira, 11 de março de 2022 17:39

Considere o algoritmo Max-Heapify:

Algorithm 9: Max-Heapify(A,i)

```
1  l = 2i;
2  r = 2i + 1;
3  if l ≤ A.heap-size and A[l] > A[i] then
4  |  largest = l;
5  end
6  else
7  |  largest = i;
8  end
9  if r ≤ A.heap-size and A[r] > A[largest] then
10 |  largest = r;
11 end
12 if largest ≠ i then
13 |  exchange A[i] with A[largest];
14 |  Max-Heapify(A,largest);
15 end
```

Considere o algoritmo Heapsort:

Algorithm 10: Heapsort(A)

```
1 Build-Max-Heap(A);
2 for i = A.length downto 2 do
3 |  exchange A[1] with A[i];
4 |  A.heap-size = A.heap-size - 1;
5 |  Max-Heapify(A,1);
6 end
```

1. (4.0 pontos) Mostre que a complexidade assintótica de Max-Heapify no pior caso em um heap com n elementos é $\Omega(\lg n)$.
2. (4.0 pontos) Mostre que a complexidade assintótica de Heapsort no pior caso em um heap com n elementos é $\Omega(n.\lg n)$.

(1.) Todas as operações realizadas nas linhas 1-11 levam tempo constante. O laço das linhas 12-15 tem uma chamada recursiva que será executada sempre que o elemento $A[i]$ não for maior do que os elementos $A[2i]$ e $A[2i+1]$. Assim, no pior caso, teremos uma chamada recursiva a partir da posição i até chegar em uma folha. Ou seja, no pior caso, temos uma chamada recursiva por nível no heap. Como o heap possui n elementos, serão $(\lg n)$ chamadas recursivas. Em cada chamada recursiva é feita uma permutação (linha 13) que tem tempo constante, e portanto o custo total de Max-Heapify no pior caso é $\Omega(\lg n)$.

(2) O algoritmo Build-Max-Heap constrói um heap de máximo em tempo linear. O pior caso de heapsort se dá quando tivermos o pior caso de Max-Heapify (linha 5) a cada iteração do laço FOR (linhas 2-6). Como o laço FOR é executado $(n-1)$ vezes, e no pior caso, Max-Heapify tem custo $\Omega(\lg i)$ $(1 \le i \le n-1)$, temos um custo total maior ou igual a $\sum_{i=1}^{n-1} \lg i = \lg((n-1)!)$ $= \Omega(n.\lg n)$.