Projeto e Análise de Algoritmos

Ordenação em Tempo Linear

Flávio L. C. de Moura*

Nesta seção veremos como informações adicionais sobre os elementos a serem ordenados podem ser utilizadas para que o processo de ordenação seja mais rápido. Como veremos, os algoritmos não são baseados na comparação de chaves, de forma que será possível ordená-los em tempo linear. Os algoritmos desta seção são conhecidos como *algoritmos de classificação*.

1 *Counting Sort*

O algoritmo *counting sort* assume que cada um dos n inteiros a serem ordenados estão no intervalo de 0 a k. Veremos que quando $k = O(n)$, a ordenação é feita em tempo $\Theta(n)$.

Algorithm 1: Counting-Sort(A, B, k)

```
1  let C[0..k] be a new array;
2  for i = 1 to k do
3  |  C[i] ← 0;
4  end
5  for j = 1 to A.length do
6  |  C[A[j]] ← C[A[j]] + 1;
7  end
8  for i = 1 to k do
9  |  C[i] ← C[i] + C[i − 1];
10 end
11 for j = A.length downto 1 do
12 |  B[C[A[j]]] ← A[j];
13 |  C[A[j]] ← C[A[j]] − 1;
14 end
```

Agora observe que o **for** das linhas 2-4 é executado em tempo $\Theta(k)$, o das linhas 5-7 em tempo $\Theta(n)$, o das linhas 8-10 em tempo $\Theta(k)$, e por fim o das linhas 11-14 em tempo $\Theta(k)$, o que perfaz um total de $\Theta(k + n)$. Assumindo que $k = O(n)$, temos que o tempo de execução de *counting sort* é $\Theta(n)$.

Definição 1.1. *Um algoritmo de ordenação é dito* estável *se não altera a posição relativa dos elementos que têm o mesmo valor.*

Exercício 1.2. *Prove que o algoritmo* counting sort *é estável.*

Exercício 1.3. *Prove que o algoritmo* merge sort *é estável.*

Exercício 1.4. *Prove que o algoritmo* insertion sort *é estável.*

2 *Radix Sort*

O algoritmo *radix sort* ordena uma sequência de inteiros com d dígitos cada, em tempo linear. Ele utiliza um algoritmo auxiliar, que precisa ser estável, para ordenar a sequência de inteiros do dígito menos significativo para o mais significativo. Utilizaremos *counting sort* como algoritmo auxiliar.

*flaviomoura@unb.br

Algorithm 2: radix-sort(A, d)

```
for i = 1 to d do
    use a stable sort to sort array A on digit i;
end
```

Lema 2.1. *Dados n números com d dígitos, que por sua vez podem assumir até k valores, radix-sort ordena corretamente estes números em tempo $\Theta(d.(n+k))$, se o algoritmo auxiliar estável tem complexidade de tempo $\Theta(n+k)$.*

Exercício 2.2. *Mostre como podemos ordenar n inteiros contidos no intervalo de 0 a n^2-1 em tempo linear, ou seja, em $O(n)$.*

Proof. Inicialmente iteramos pela lista dos números, convertendo cada um deles para a base n. Depois aplicamos o algoritmo *radix sort* sobre a lista, utilizando o *counting sort* como o algoritmo de ordenação dos dígitos da lista de números.

Dada a nova base, cada número possuirá 2 dígitos, uma vez que $\log_n(n^2) = 2$, onde cada dígito estará em um intervalo entre 0 e $n-1$. Sabendo disso, vemos que *radix sort* ordernará n números de 2 dígitos, usando o *counting sort* em um intervalo de 0 à $n-1$, resultando em uma complexidade de $O(2(n+n)) = O(n)$ □

Exercício 2.3. *Mostre como podemos ordenar n inteiros contidos no intervalo de 0 a n^3-1 em tempo linear, ou seja, em tempo $O(n)$.*

3 Leitura complementar:

1. [1] (Capítulo 8)
2. [2] (Capítulo 7, Secão 7.1)

Referências

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[2] A. V. Levitin. *Introduction to the Design and Analysis of Algorithms, Third Edition.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2012.