

Semana 4: Ordenação – Counting Sort e Radix Sort

Prof. Dr. Juliano Henrique Foleis

Estude com atenção os vídeos abaixo. Os exercícios servem para ajudar na fixação do conteúdo e foram escolhidos para complementar o material básico apresentado nos vídeos. Quando o exercício pede que crie ou modifique algum algoritmo, sugiro que implemente-o em linguagem C para ver funcionando na prática. O único exercício que é necessário entregar está descrito na Seção “Atividade Para Entregar”.

Vídeos

[Counting Sort \(Ordenação Por Contagem\)](#)

[Radix Sort \(Ordenação por Dígitos\)](#)

Exercícios

1. Cronometragem do CountingSort e RadixSort. Neste exercício serão ordenados vetores da seguinte estrutura:

```
typedef struct Info {  
    int chave;  
    int dado;  
} Info;
```

a. Escreva uma função *Info* random_info_vector(int n, int max, int seed)* que retorne um vetor com n elementos com chaves e dados aleatórios entre 0 e max . Seed é a semente usada para iniciar o gerador de números aleatórios.

b. Execute CountingSort e Radixsort em um vetor aleatório gerado com a função *Info* random_info_vector(int n, int max, int seed)* com $n = 1000, 10000, 100000, 500000$ e 1000000 , $max = n * 100$ e $seed = 42$. Anote o tempo de execução para cada n . Ordene os mesmos vetores com MergeSort, QuickSort e HeapSort e compare os tempos de execução.

2. Altere o CountingSort apresentado no vídeo para operar ordenar vetores que também contenham chaves negativas. **DICA:** você pode transformar as chaves todas em inteiros positivos! Só não esqueça de voltar as chaves para os valores corretos após a ordenação!

3. No vídeo do Counting Sort (em 9:34) eu mencionei que se o vetor sendo ordenado fosse apenas um vetor de inteiros, a ordenação já estaria concluída naquele momento. Altere o algoritmo apresentado no vídeo para ordenar um vetor de inteiros sem que seja necessário computar a soma de prefixos. Compare o tempo desta versão do Counting Sort com RadixSort, MergeSort, QuickSort e RadixSort para ordenar vetores aleatórios gerados usando *int* random_vector(int n, int max, int seed)* com $n = 1000, 10000, 100000, 500000, 1000000$, $max = n * 100$ e $seed = 0$.

Atividade para Entregar

A atividade a seguir é para ser feita individualmente e entregue via Moodle no tópico da Semana 4. A data-limite para entrega é dia 2/9/2020 às 23:55. Em caso de cópia as atividades dos participantes serão

desconsideradas.

Descrição da Atividade

Internamente os computadores representam números inteiros em formato binário. Em outras palavras, os números são armazenados em base 2. Por causa disso, a divisão por potências de 2 pode ser realizada com instruções de deslocamento de bits (*bit shifting*). Desta forma, a fórmula de obtenção dos dígitos pode ser reescrita em C na base 2 como: $(N \gg pos) \& 1$, tal que N é o número, pos é a posição do bit a ser copiado (0 é a posição do bit menos significativo, à direita) e $\&$ é a operação binária AND.

1. Na maioria das arquiteturas de computadores modernas, as instruções de deslocamento de bits são muito mais rápidas que as instruções de divisão. Para verificar se a ordenação na base 2 pode ser mais rápida que na base 10 no Radix Sort, reimplente o Radix Sort em base 2 utilizando essa nova função para obter os dígitos durante o Counting Sort. Use inteiros de 32-bits para armazenar cada elemento.

2. Compare o tempo de execução do Radix Sort na base 2 e do Radix Sort na base 10 ordenando vetores aleatórios gerados usando `int* random_vector(int n, int max, int seed)` com $n = 1000, 10000, 100000, 500000, 1000000$, $max = n * 100$ e $seed = 0$. Anote os resultados na Tabela abaixo em milissegundos (*ms*).

	1000	10000	100000	500000	1000000
RadixSort (Base 2)					
RadixSort (Base 10)					

3. Analisando os resultados obtidos, qual versão foi mais rápida? Por quê?

Você deve Entregar

Entregue em formato .zip os arquivos a seguir:

- Os arquivos-fonte desenvolvidos nos itens 1 e 2. Faça um *Makefile* para compilar o seu programa. Modularize conforme julgar necessário.
- A Tabela preenchida no item 2 e a resposta da pergunta 3 em um *pdf*. Sua resposta do item 3 deve ter pelo menos 3 linhas.

Por favor entregue como especificado acima!

A data-limite para entrega é dia 30/3/2021 às 23:55.

BONS ESTUDOS!