CI1056: Algoritmos e Estruturas de Dados II

Prof. Dr. Marcos Castilho

Departamento de Informática/UFPR

7 de dezembro de 2020

Resumo

Ordenação por contagem

Objetivos da aula

- Apresentar o algoritmo de ordenação por contagem, ou *countingsort*

Motivação

- Sabemos que o limite inferior para ordenação por comparação entre elementos de um conjunto é $n.log_2(n)$ no pior caso
- Porém, para alguns problema não é necessário comparar os elementos, são problemas especiais, cujos dados têm alguma propriedade que pode ser explorada
- É o caso do *counting sort*

Ordenação por contagem

counting_sort (v, w, n)

- Instância: (v, n), onde v é um vetor de tamanho n e seus elementos têm a propriedade de serem números inteiros na faixa $[1..k]$, para algum inteiro k, podendo haver repetição entre os números
- Resposta: retorna um (w, n) de forma que w é vetor ordenado

O princípio da ordenação por contatgem

- Determina-se, para cada elemento de entrada x, o número de elementos que são menores que x e usa esta informação para inserir x diretamente na sua posição final no vetor de saída
- Por exemplo, se 17 elementos forem menores que x, então x deve estar na posição 18 do vetor de saída
- Tem que se tomar cuidado com os elementos repetidos

Vetor auxiliar

- O algoritmo usará um vetor auxiliar para ajudar no processo
- Seja $c[0..k]$ este vetor auxiliar

Exemplo

- Seja o vetor $v = [2, 5, 3, 0, 2, 3, 0, 3]$ contendo 8 elementos cujos valores não inteiros não negativos não maiores do que 5

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3

- Seja $c[0..5]$ um vetor auxiliar para contar quantos elementos iguais ao índice $i \in [0..5]$ ocorrem no vetor v
- Observe que o 0 ocorre 2 vezes, o 1 não ocorre, o 2 ocorre 2 vezes, o 3 ocorre 3 vezes, o 4 não ocorre e o 5 ocorre 1 vez

	0	1	2	3	4	5
c	2	0	2	3	0	1

Como obter o vetor c

- É bem simples obter c, já que os elementos do vetor são inteiros e podem ser usados como índices de vetor
- Tem que zerar c antes e depois incrementar o contador do respectivo índice conforme o elemento do vetor v
- Após esta etapa, c contém a contagem dos elementos de v

```
Seja c[0..k] um novo vetor
para i 0 ate k
    c[i]= 0
para i de 1 ate n
    c[v[i]]= c[v[i]] + 1
```

Exemplo

	0	1	2	3	4	5
c	2	0	2	3	0	1

- Agora contamos quantos elementos são menores ou iguais a i, para todo i

```
para i de 1 ate k
    c[i]= c[i] + c[i-1]
```

	0	1	2	3	4	5
c	2	2	4	7	7	8

Última etapa

- Regressivamente, colocamos os elementos no lugar deles em w
- E reduzimos a contagem, pois a cada iteração 1 elemento foi tratado

```
para i de n ate 1
    w[c[v[i]]]= v[i]
    c[v[i]]= c[v[i]] - 1
```

Vejamos passo a passo

- na primeira iteração $i = 8$
- `w[c[v[8]]] = w[c[3]] = w[3]`
- então `w[3]= v[8]= 3`
- e decrementamos `c[v[8]] = c[3]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w							3	

	0	1	2	3	4	5
c	2	2	4	6	7	8

Vejamos passo a passo

- na segunda iteração $i = 7$
- `w[c[v[7]]] = w[c[0]] = w[2]`
- então `w[2]= v[7]= 0`
- e decrementamos `c[v[7]] = c[0]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w		0					3	

	0	1	2	3	4	5
c	1	2	4	6	7	8

Vejamos passo a passo

- na terceira iteração $i = 6$
- `w[c[v[6]]] = w[c[3]] = w[6]`
- então `w[6]= v[6]= 3`
- e decrementamos `c[v[6]] = c[3]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w		0				3	3	

	0	1	2	3	4	5
c	1	2	4	5	7	8

Vejamos passo a passo

- na quarta iteração $i = 5$
- `w[c[v[5]]] = w[c[2]] = w[4]`
- então `w[4]= v[5]= 2`
- e decrementamos `c[v[5]] = c[2]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w		0		2		3	3	

	0	1	2	3	4	5
c	1	2	3	5	7	8

Vejamos passo a passo

- na quinta iteração $i = 4$
- `w[c[v[4]]] = w[c[0]] = w[1]`
- então `w[1]= v[4]= 0`
- e decrementamos `c[v[4]] = c[0]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w	0	0		2		3	3	

	0	1	2	3	4	5
c	0	2	3	5	7	8

Vejamos passo a passo

- na sexta iteração $i = 3$
- `w[c[v[3]]] = w[c[3]] = w[5]`
- então `w[5]= v[3]= 3`
- e decrementamos `c[v[3]] = c[3]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w	0	0		2	3	3	3	

	0	1	2	3	4	5
c	0	2	3	4	7	8

Vejamos passo a passo

- na sétima iteração $i = 2$
- `w[c[v[2]]] = w[c[5]] = w[8]`
- então `w[8]= v[2]= 5`
- e decrementamos `c[v[2]] = c[5]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w	0	0		2	3	3	3	5

	0	1	2	3	4	5
c	0	2	3	4	7	7

Vejamos passo a passo

- na oitava e última iteração $i = 1$
- `w[c[v[1]]] = w[c[2]] = w[3]`
- então `w[3]= v[1]= 2`
- e decrementamos `c[v[1]] = c[2]`

	1	2	3	4	5	6	7	8
v	2	5	3	0	2	3	0	3
w	0	0	2	2	3	3	3	5

	0	1	2	3	4	5
c	0	2	2	3	7	7

O algoritmo *counting sort*

```
Seja c[0..k] um novo vetor
para i 0 ate k
    c[i]= 0
para i de 1 ate n
    c[v[i]]= c[v[i]] + 1
para i de 1 ate k
    c[i]= c[i] + c[i-1]
para i de n ate 1
    w[c[v[i]]]= v[i]
    c[v[i]]= c[v[i]] - 1
```

Observações

- Note que não há comparações entre elementos do vetor
- Só existem atribuições
- O custo do algoritmo é:
  - $k + 1$ atribuições para zerar o vetor c
  - n atribuições para a primeira contagem de c
  - k atribuições para determinar o número de elementos menores ou iguais a i no vetor c
  - $2n$ atribuições para construir o vetor ordenado w
- Total: $3n + 2k + 1$. Quando k é linear em n, a ordenação é linear
- A ordenação é estável

Fim do tópico

- O conteúdo desta aula está no livro Cormen, Leiserson, Rivest e Stein, no capítulo 8, seção 8.2.

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`