CI1056: Algoritmos e Estruturas de Dados II

Prof. Dr. Marcos Castilho

Departamento de Informática/UFPR

8 de dezembro de 2020

Resumo

Ordenação por baldes

Objetivos da aula

- Apresentar um algoritmo de ordenação por baldes, conhecido como *bucket sort*

Motivação

- Existe uma categoria de problemas de ordenação na qual ainda podemos melhorar a complexidade teórica
- Para esta categoria não podemos aplicar nem o *counting sort* nem o *radix sort*
- Nesta categoria, para poder fazer melhor do que $n.log_2(n)$ é preciso também contar com alguma peculiaridade nos dados do conjunto

Propriedade dos conjuntos

- Os elementos do conjunto são números reais $x \in [0..1)$
- Eles devem estar distribuídos uniformemente!

Ordenação por baldes

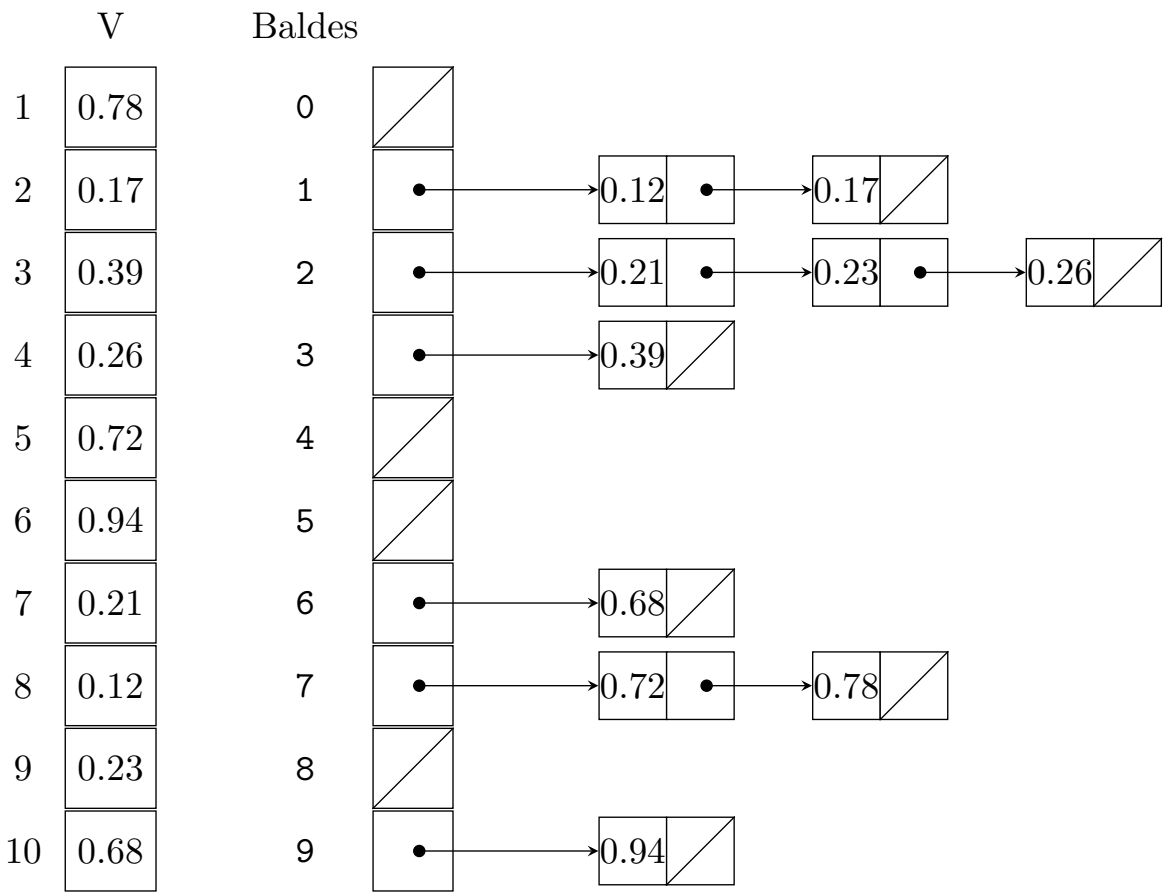
bucket_sort (v, n)

- Instância: (v, n), onde v é um vetor de tamanho n e seus elementos têm a propriedade de serem números reais *uniformemente* distribuídos no intervalo semi-aberto [0..1)
- Resposta: retorna um (v, n) de forma que v é vetor ordenado

O princípio da ordenação por baldes

- Dado que os elementos estão uniformemente distribuídos
- Vamos dividir os elementos em exatamente n subintervalos de tamanhos iguais, denominados de *baldes*
- Vamos distribuir os n elementos nestes n baldes segundo uma operação matemática simples
- Esta operação matemática vai colocar os elementos nos baldes de maneira que se espera que cada balde fique com poucos elementos
- Assim, é possível ordenar cada balse a um custo baixo e em seguida concatenar os elementos dos baldes na ordem certa para obtermos o vetor ordenado

Exemplo


V
Baldes
1
0.78
2
0.17
3
0.39
4
0.26
5
0.72
6
0.94
7
0.21
8
0.12
9
0.23
10
0.68
0
1
2
3
4
5
6
7
8
9
0.12
0.17
0.21
0.23
0.26
0.39
0.68
0.72
0.78
0.94

Como funciona?

- A implementação clássica do *bucket sort* usa listas ligadas para cada balde
- Mas também pode ser usado um vetor
- Como os elementos estão uniformemente distribuídos e portanto cada balde contém uma pequena lista de elementos, pode-se usar o algoritmo *insertion sort* para ordenar cada balde
- A própria distribuição entre os baldes já é uma pré-ordenação

A conta que fizemos no exemplo

- No exemplo acima os números foram distribuídos nos baldes segundo a conta
- $balde[\lfloor nV[i] \rfloor] = V[i]$
- Isto é, cada elemento $V[i]$ é colocado no balde bastando multiplicar o valor de $V[i]$ por n e tomar o piso

Desta maneira

- O balde 0 vai conter os elementos do intervalo [0.0..0.1)
- O balde 1 vai conter os elementos do intervalo [0.1..0.2)
- O balde 2 vai conter os elementos do intervalo [0.2..0.3)
- e assim por diante, até:
- O balde 9 vai conter os elementos do intervalo [0.9..1.0)

Observações

- É claro, no exemplo, que alguns baldes estão vazios
- Outros têm 1 elemento
- Outros tem 2 ou 3 elementos
- Mas como são n elementos divididos em n baldes
- E como os elementos estão uniformemente distribuídos
- Então se espera realmente que cada balde tenha bem poucos elementos
- Por isso o *insertion sort* é o algoritmo escolhido

Observações

- Obviamente, se os elementos não estiverem uniformemente distribuídos será um desatre
- Algum dos baldes pode ter próximo de n elementos
- E o *insertion sort* é quadrático no pior caso
- Felizmente você é um bom programador e não vai usar este método sobre os dados errados!

O algoritmo

```
bucketsort (v, n)
1    Seja B[0..n-1] um vetor auxiliar para os baldes
2    para i de 0 ate n-1
3        faca B[i] uma lista vazia
4    para i de 1 ate n
5        insira a[i] no balde B[piso (n * A[i])]
6    para i de 0 ate n-1
7        ordene a lista B[i] usando o insertion sort
8    concatene as listas B[0], B[1], ... , B[n-1] em ordem
```

Análise preliminar

- É evidente que as linhas 1, 2-3, 4-5 e 8 têm custo linear
- A linha 6 é um laço de n iterações
  - Obviamente, o *bucket sort* só será linear se o *insertion sort*, neste caso, tiver custo constante
  - E tem!

Prova

- Lamentavelmente a prova não será apresentada
- Ela está na seção 8.4 do livro do Cormen, são duas páginas
- A prova exige conhecimentos de probabilidade que vocês ainda não têm
- Evidentemente a demonstração explora o fato dos elementos do vetor estarem uniformemente distribuídos

Fim do tópico

- O conteúdo desta aula está no livro Cormen, Leiserson, Rivest e Stein, no capítulo 8, seção 8.4

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/`