CI1056: Algoritmos e Estruturas de Dados II

Prof. Dr. Marcos Castilho

Departamento de Informática/UFPR

8 de dezembro de 2020

Resumo

Ordenação digital (ou ordenação de *strings*)

Objetivos da aula

- Apresentar um algoritmo de ordenação digital, conhecido como *radix sort*

Motivação: alguns problemas interessantes

- Suponha que se queira contar o número de diferentes placas de veículos que passam por uma estrada
- Suponha que se queira contar o número de diferentes IP's que chegam em um provedor de Internet (IP é o endereço de um computador)
- Suponha que você queira ordenar um baralho para deixá-lo na forma como ele veio quando você o comprou
- Suponha que você queira ordenar datas, por ano depois por mês e finalmente por dias

Motivação

- O que estes problemas têm em comum?
  - Os itens a serem ordenados podem ser considerados como *strings*
  - Além disso, as *strings* têm o mesmo tamanho!

Motivação

- No Brasil em 2020 as placas de carro são formadas por letras e números, mas todas elas têm exatamente 7 caracteres (AAA1212, BBB4637, ...)
- Os endereços IP (V4) na internet são formados por exatamente 4 números que podem ser representados por 1 byte (200.236.12.48, 192.168.1.5, ...), cada um destes números pode variar entre 0 e 255
- Uma carta de baralho pode ser representada por seu naipe (O, C, E, P) seguido de seu número (1 a 13)
- Uma data por ser colocada no formato AAAA/MM/DD, sempre 10 caracteres

Motivação

- Claro que pode se aplicar um método de ordenação clássico
- Mas a ordenação digital é mais eficiente!
- Exploramos o fato de que as chaves têm exatamente o mesmo tamanho

Algoritmos conhecidos

- Existem basicamente dois tipos de algotitmos:
  - LSD: *Least Significant Digit first*, ou seja, primeiro o dígito menos significativo
  - MSD: *Most Significant Digit first*, ou seja, primeiro o dígito mais significativo
- Nós veremos o primeiro e deixaremos o segundo como exercício adicional

Ordenação digital LSD

radix_sort (v, n, k)

- Instância: (v, n), onde v é um vetor de tamanho n e seus elementos têm a propriedade de serem *strings* de mesmo tamanho, digamos k, para algum inteiro k
- Resposta: retorna um (v, n) de forma que v é vetor ordenado

O princípio da ordenação digital

- Se todas as chaves são *strings* de tamanho k, ordena-se em k passos:
  - primeiro ordenamos pelo dígito menos significativo
  - depois pelo segundo menos significativo
  - e assim por diante, até
  - por último, ordena-se pelo dígito mais significativo
- Este método funciona desde que a ordenação seja *estável*

Exemplo

- Seja o seguinte vetor contendo $n = 8$ placas de veículos, todas elas contendo 7 caracteres:

PGC4938
IYE2230
CIO3720
ICK1750
OHV1845
JZY4524
CIO3720
OHV1845

Exemplo

- Primeiro, observamos o caractere menos significativo e ordenamos por este valor (a ordenação deve ser estável):

PGC4938
IYE2230
CIO3720
ICK1750
OHV1845
JZY4524
CIO3721
OHV1846

IYE2230
CIO3720
ICK1750
CIO3721
JZY4524
OHV1845
OHV1846
PGC4938

Exemplo

- Agora observamos o segundo caractere menos significativo e ordenamos por este valor (a ordenação deve ser estável):

IYE2230
CIO3720
ICK1750
CIO3721
JZY4524
OHV1845
OHV1846
PGC4938

CIO3720
CIO3721
JZY4524
IYE2230
PGC4938
OHV1845
OHV1846
ICK1750

Exemplo

- Agora observamos o terceiro caractere:

CIO3720
CIO3721
JZY4524
IYE2230
PGC4938
OHV1845
OHV1846
ICK1750

IYE2230
JZY4524
CIO3720
CIO3721
ICK1750
OHV1845
OHV1846
PGC4938

Exemplo

- Agora observamos o quarto caractere:

IYE2230
JZY4524
CIO3720
CIO3721
ICK1750
OHV1845
OHV1846
PGC4938

ICK1750
OHV1845
OHV1846
IYE2230
CIO3720
CIO3721
JZY4524
PGC4938

Exemplo

- Agora observamos o quinto caractere:

ICK1750
OHV1845
OHV1846
IYE2230
CIO3720
CIO3721
JZY4524
PGC4938

PGC4938
IYE2230
ICK1750
CIO3720
CIO3721
OHV1845
OHV1846
JZY4524

Exemplo

- Agora observamos o penúltimo caractere:

PGC4938
IYE2230
ICK1750
CIO3720
CIO3721
OHV1845
OHV1846
JZY4524

ICK1750
PGC4938
OHV1845
OHV1846
CIO3720
CIO3721
IYE2230
JZY4524

Exemplo

- Finalmente, o último caractere:

ICK1750
PGC4938
OHV1845
OHV1846
CIO3720
CIO3721
IYE2230
JZY4524

CIO3720
CIO3721
ICK1750
IYE2230
JZY4524
OHV1845
OHV1846
PGC4938

Como funciona?

- Intuitivamente, se os caracteres que não foram examinados para um par são iguais, qualquer diferença entre as chaves é restrita aos caracteres que já foram examinados
- Então as chaves já foram ordenadas corretamente e vão permanecer por causa da estabilidade

O algoritmo *radix sort*

- Sem apresentar código, observe que no processo de ordenação temos um único caractere por vêz
- Suponha que eles são caracteres válidos da tabela ASCII, isto é, são 256 caracteres
- Logo, podemos adaptar o *counting sort* da aula passada, já que ele é um algoritmo *estável*
- Basta ter uma função que considera um dos dígitos de uma determinada *string*

Observações

- Seja n o tamanho do vetor v
- Seja k o tamanho das *strings*
- O *counting sort* ordena em tempo linear (n)
- Logo, o *radix sort* ordena em tempo kn
- Em geral, $k << n$, então a ordenação é muito eficiente!

Fim do tópico

- O conteúdo desta aula está no livro Cormen, Leiserson, Rivest e Stein, no capítulo 8, seção 8.3 e no livro Sedgewick e Wayne, seção 5.1

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/`