1.Radix sort

Grupo 2:

- Bruno Lírio
- Pedro Paulo Oliveira
- Thales Veras

2.Motivação

2.1Definição do problema

2.1.1Descrição Informal

O radix sort ou ordenação digital surgiu para resolver o problema de ordenação de cartões perfurados. Esses cartões possuíam um número c de colunas na qual era possível uma perfuração em p posições. Assim, eram examinadas as perfurações coluna a coluna e de acordo com a posição da perfuração os cartões eram distribuídos de

forma iterativa em p caixas c vezes, no final tínhamos os cartões ordenados. É importante observar que enquanto ocorre a ordenação, ao longo do processo iterativo não se pode mudar a ordem das caixas.

2.1.2Definição Formal do Problema

Entrada: Temos um conjunto de números

Questão: Como podemos ordenar esses números (neste caso, sem comparações)?

Saída: Conjunto de números ordenado

2.2Exemplo de instância

Como podemos ordenar o seguinte conjunto de números sem a necessidade de compará-los?

176, 44, 80, 9, 15,76

3.1Descrição da ideia do algoritmo

O algoritmo funciona analisando o conjunto de números a serem ordenados,os algarismos de posição menos significativa para a posição mais significativa, num processo iterativo para cada posição, ordenando provisoriamente cada número a cada passo da iteração e chegando ao final do processo com os elementos de fato ordenados.

3.2 Exemplo de solução da instância

1º passo da iteração:

176, 44, 80, 9, 15,76 -> 176,044,080,009,015,076

0 -> 080

1

2

3

4 -> 044

5 -> 015

6 -> 176,076

7

8

9 -> 009

2ª passo da iteração

080,044,015,176,076,009

0 -> 009

1 -> 015

2

3

4 ->044

5

6

7 -> 176,076

8 -> 080

9 ->

3ª passo da iteração

009,015,044,176,076,080

0 -> 009,015,044,076,080	6
1 -> 176	7
2	8
3	9
4	
5	

saída: 009,015,044,076,080,176

3.3Descrição formal do algoritmo

```
para i = 1..d faça
    para j = 1..n faça
        k:= i-ésimo dígito menos significativo da
representação de
            L[j].chave na base b
        Fk <= L[j]
    j := 1
    para k = 0..b - 1 faça
        enquanto Fk (diferente de) Ø faça
            L[j] <= Fk
            j := j + 1
```

3.4Detalhar as etapas da aplicação técnica

O radix-sort faz uso de filas que auxiliam no armazenamento intermediário dos elementos a serem ordenados, em cada iteração, bem como das operações de inserção e remoção.

3.5 Implementação do radix-sort em C

```
void RADIXSORT(int elemento[], int digito) {
    int i;
    int b[digito];
    int maior = elemento[0];
    int exp = 1;

    for (i = 0; i < digito; i++) {
        if (elemento[i] > maior)
            maior = elemento[i];
    }

    while (maior/exp > 0) {
        int bucket[10] = { 0 };
        for (i = 0; i < digito; i++)
            bucket[(elemento[i] / exp) % 10]++;
        for (i = 1; i < 10; i++)
            bucket[i] += bucket[i - 1];
        for (i = digito - 1; i >= 0; i--)
            b[--bucket[(elemento[i] / exp) % 10]] = elemento[i];
        for (i = 0; i < digito; i++)
            elemento[i] = b[i];
        exp *= 10;
```

4.Análise do Algoritmo

4.1Prova de corretude

Considerando uma entrada qualquer de n números e d dígitos, ao final da primeira iteração temos que o algoritmo terá ordenado os n números em relação ao dígito menos significativo, assim por indução, temos que nas próximas iterações o algoritmo terá ordenado os d-1 dígitos restantes, dos n números, o que resultará na ordenação dos n números.

4.2Demonstração da complexidade

Quando cada dígito está numa base na qual seu valor varia de 0 a b-1 (exemplo, de 0 a 9), temos que cada passagem sobre os n números de d dígitos, para sua ordenação, leva o tempo $\Theta(n + b)$, como são d passagens, temos $\Theta(d(n + b))$ mas, d é o número de dígitos dos números que é uma constante portanto,

$d*\Theta(n + b) = \Theta(n + b) = máx(\Theta(n),\Theta(b))$ que indica que o radix-sort pode ser executado em tempo linear.

4.Conclusão e discussões

4.1 Vantagens e desvantagens do radix-sort

-é um dos algoritmos de ordenação mais rápidos

-simples

-intuitivo

-pode ser bem custoso se for necessário avaliar elementos com diferentes números de dígitos

-pode requerer diferentes implementações dependendo do tipo de dado

4.2É possível fazer melhor?

Se considerarmos ordenações nas quais seja possível fazer comparações o merge-sort talvez pudesse ser uma escolha de menor tempo de execução.

5.Referências Bibliográficas

1.Radix Sort. (2003). WIKIPEDIA. ***Site.*** Disponível em: <http://pt.wikipedia.org/wiki/Radix_sort>. Acesso em: 01 out. 2013.

2.CORMEN, Thomas H.. **Algoritmos: teória e prática.** Rio de Janeiro: Editora Campus, 2002.

3.SZWARCFITER, Jayme Luiz **Estruturas de Dados e Seus Algoritmos.** Rio de Janeiro Editora LTC, 2010.