Aula RadixSort

Referência: Sedgewick, 1988
Capítulo 10

Radix Sorts

- Algumas vezes, a comparação entre chaves pode ser uma operação muito cara
- Existem algoritmos de ordenação sem comparação de chaves
- Radix Sorts: métodos que fazem uso da representação digital dos registros:
  - Não apenas comparam chaves
  - Processam e comparam partes de chaves
  - Tratam chaves como números representados em uma base númerica M

Analogia – Ordenação de Cartas

- Considere um conjunto de cartas cada uma contendo 3 dígitos decimais (0 a 999), ou seja, M =10
  - Separa em 10 pilhas: 1 para números < 100; outro para números entre 100 e 199, outra para números entre 200 e 299 e assim por diante.
  - Lida com cada pilha individualmente fazendo o mesmo para o próximo dígito. Por exemplo, separa a pilha de todos cujo 1o. dígito é 0 em pilhas de acordo com o 2o. dígito, ou seja, de 000 a 009; de 010 a 019; de 020 a 029 e assim por diante.

Idéia geral

- Usa a base M = 2
- Chaves cujo bit mais a esquerda é 0, vem antes que chaves cujo bit é 1
- Repetindo-se isso para todos os bits de forma adequada é possível ordenar

Em PASCAL

- Difícil manipulação de números binários
- Para separar bit desejado usa-se as operações SHIFT e AND
- Em PASCAL simula estas operações com operadores div e mod:
  - Desloque (SHIFT) x de k = $(x \text{ div } 2^k)$
  - Zere todos os bits menos os j bits mais à direita de x = $(x \text{ mod } 2^j)$

Simulando SHIFT e AND

- **function bits (x, k, j: integer): integer**, combina as 2 operações e retorna os j bits que aparecem k bits a partir da direita de x (ou seja, calcula $((x \text{ div } 2^k) \text{ mod } 2^j)$)
- Como no algoritmo Radix Exchange Sort queremos apenas um bit que aparecem k bits a partir da direita de x então usaremos:

  function bits (x, k: integer): integer;

  que considera sempre j=1.

Algoritmo Radix Exchange Sort

- Algoritmo analisa os bits da esquerda para a direita
- Funcionamento similar ao do Quicksort, mas a partição é feita comparando-se bits ao invés de chaves
- Chamadas recursivas ordenando os subvetores pelo bit i-1

Programa Radix Exchange Sort

```
procedure RadixExchangeSort(var A: Vetor; esq, dir, b: integer);
var i, j: Indice;
w: Item;
begin
  if (esq < dir) and (b >= 0) then begin
    i := Esq; j := Dir;
    repeat
      while (Bits(A[i].Chave,b) = 0) and (i<j) do i:=i+1;
      while (Bits(A[j].Chave,b) = 1) and (i<j) do j:=j-1;
      w := A[i]; A[i] := A[j]; A[j] := w;
    until i = j;
    if (bits(A[j].chave,b) = 0) then j := j + 1;
    RadixExchangeSort(A, Esq, j-1, b-1);
    RadixExchangeSort(A, j, Dir, b-1);
  end;
end;
```

Análise do Algoritmo

- Ordem de complexidade: N x b
- Onde:
  - b é o número de bits; e
  - N o número de elementos e.
- No pior caso, onde se tenha um registro para cada combinação possível de bits, b será $\log_2 N$.