Ordenações - Parte 3
Ordenações Lineares

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Ordenação sem comparação

- Vimos vários algoritmos de ordenação que se baseiam em comparações para ordenar os vetores
  - Esses algoritmos são limitados por O(n log n)
- Existe uma categoria para aqueles algoritmos que não se baseiam em comparação e, portanto, podem passar deste limite
  - Eles usam mais memória e é necessário saber o alcance dos valores, tamanho de chaves ou de dígitos

Non-comparison sorts

Name	Best	Average	Worst	Memory	Stable	$n \ll 2^k$	Notes
Pigeonhole sort	—	$n + 2^k$	$n + 2^k$	2^k	Yes	Yes	
Bucket sort (uniform keys)	—	$n + k$	$n^2 \cdot k$	$n \cdot k$	Yes	No	Assumes uniform distribution of elements from the domain in the array.[17]
Bucket sort (integer keys)	—	$n + r$	$n + r$	$n + r$	Yes	Yes	If r is $O(n)$, then average time complexity is $O(n)$.[18]
Counting sort	—	$n + r$	$n + r$	$n + r$	Yes	Yes	If r is $O(n)$, then average time complexity is $O(n)$.[17]
LSD Radix Sort	—	$n \cdot \frac{k}{d}$	$n \cdot \frac{k}{d}$	$n + 2^d$	Yes	No	$\frac{k}{d}$ recursion levels, 2^d for count array.[17][18]
MSD Radix Sort	—	$n \cdot \frac{k}{d}$	$n \cdot \frac{k}{d}$	$n + 2^d$	Yes	No	Stable version uses an external array of size n to hold all of the bins.
MSD Radix Sort (in-place)	—	$n \cdot \frac{k}{1}$	$n \cdot \frac{k}{1}$	2^1	No	No	d=1 for in-place, $k/1$ recursion levels, no count array.
Spreadsort	n	$n \cdot \frac{k}{d}$	$n \cdot \left(\frac{k}{s} + d\right)$	$\frac{k}{d} \cdot 2^d$	No	No	Asymptotic are based on the assumption that $n \ll 2^k$, but the algorithm does not require this.
Burstsort	—	$n \cdot \frac{k}{d}$	$n \cdot \frac{k}{d}$	$n \cdot \frac{k}{d}$	No	No	Has better constant factor than radix sort for sorting strings. Though relies somewhat on specifics of commonly encountered strings.
Flashsort	n	$n + r$	n^2	n	No	No	Requires uniform distribution of elements from the domain in the array to run in linear time. If distribution is extremely skewed then it can go quadratic if underlying sort is quadratic (it is usually an insertion sort). In-place version is not stable.
Postman sort	—	$n \cdot \frac{k}{d}$	$n \cdot \frac{k}{d}$	$n + 2^d$	—	No	A variation of bucket sort, which works very similar to MSD Radix Sort. Specific to post service needs.

Fonte:
https://en.wikipedia.org/wiki/Sorting_algorithm#Non-comparison_sorts

Ordenação por distribuição

- Nós vamos estudar hoje particularmente aqueles que alguns autores chamam de ordenação por distribuição (*distribution sort*)
  - Qualquer algoritmo que os dados são distribuídos a partir da entrada em várias estruturas intermediárias que então são unidas e colocadas como saída
- Vamos ver:
  - *Counting Sort, Radix Sort, Bucket Sort*

Counting Sort

Counting Sort

- Focado em contar a quantidade de objetos que tem a mesma chave
  - Usa aritmética para determinar a posição de cada valor de chave na sequência de saída
- É linear no número de itens e a diferença entre o número máximo e mínimo de valores de chaves
  - Bom para uso em situações que a variação do valor de chaves não é muito maior que o número de itens

Counting Sort

- O (n + k)
  - *k* é a variação entre o maior e menor valor possível da chave (positiva)

Radix Sort

Radix Sort

- Distribui elementos em “baldes” (*buckets*) de acordo com sua *radix*
  - Sistema numérico posicional
  - A *radix* ou base é o número de dígitos únicos, incluindo o zero, usado para representar os números
  - Para o sistema decimal, 10 (0 a 9)

Radix Sort

- Para elementos com mais de um dígito significativo
  - O processo de colocar em baldes é repetido para cada dígito
    - A ordem do passo anterior é preservada
- Pode ser aplicado em dados que podem ser ordenados lexicograficamente
  - Inteiros, palavras, cartas de baralho, cartas de correio

Radix Sort

- O(w*n)
  - *w* é o número de bits necessários para representar cada chave

Bucket Sort

Bucket Sort

- Distribui elementos de um vetor em um número de “baldes”
- Cada “balde” é ordenado individualmente
  - Usando outro algoritmo de ordenação ou chamando o *bucket sort* recursivamente
- Depende do algoritmo usado para ordenar cada “balde”, o número de “baldes” e se a entrada está distribuída uniformemente

Bucket Sort

- O(n^2) no pior caso
- O($n+(n^2/k)+k$) no caso médio
  - k é o número de “baldes”
  - Quando *k* é próximo de *n*, chega a O(n)

Referências

Referências

- CORMEN, T. H.; RIVEST, R. L.; LEISERSON, C. E.; STEIN, C.. Algoritmos: teoria e prática. Elsevier, 2012.
- https://en.wikipedia.org/wiki/Counting_sort
- https://www.geeksforgeeks.org/counting-sort/
- https://en.wikipedia.org/wiki/Radix_sort
- https://www.geeksforgeeks.org/radix-sort/
- https://en.wikipedia.org/wiki/Bucket_sort
- https://www.geeksforgeeks.org/bucket-sort-2/
- https://www.cs.usfca.edu/~galles/visualization/Algorithms.html