Tipo de bolha

Varra seus dados trocando itens adjacentes se eles estiverem no caminho errado. Uma melhoria de fator de 2 pc ser feita fazendo varreduras sucessivas ficarem mais curtas. A complexidade de tempo é (n-1)(n-2)/2 comparaç e esperamos (em média) fazer metade desse número de trocas. Se não trocar itens iguais, fica estável.*

Vem em dois sabores básicos. Um varre os dados até saber que não pode haver problemas restantes (n-1 varreduras, como acima), o outro define um sinalizador se os itens foram trocados e varre até que o sinalizador permaneça falso. O último pára mais cedo se os dados foram quase classificados para começar, mas carrega un pequeno custo extra associado a ter e definir o sinalizador.

BubbleSort é fácil de entender, fácil de codificar, mas ineficiente. Um Otimizador Realmente Agressivo hipotétic reconhece automaticamente as implementações de graduação do tipo de bolha e as substitui por algo melhor.

Separando as ideias de "encontrar o maior" e "mover o maior" dá InsertionSort e SelectionSort , cujas velocida comparativas dependem de qual é mais caro, comparações ou trocas. Veja essas páginas para mais detalhes.

*BubbleSort nunca é mais rápido que InsertionSort , desde que InsertionSort não use **uma** pesquisa binária.*

- *BubbleSort **é** mais rápido que InsertionSort em uma matriz de 3 elementos.*

É importante lembrar que o Bubble Sort não é um algoritmo 100% inútil - se você tiver uma sequência de objeto que gostaria de manter ordenados que ocasionalmente é perturbada pelo aumento ou diminuição do valor de un dois objetos, o Bubble Sorting é uma coisa boa. Considere o buffer Z de um campo de objetos principalmente imóveis. Digamos que um ou dois objetos se aproximem ou se afastem do observador - a classificação seria simplesmente trocar alguns slots adjacentes, para o que o Bubble Sort é ideal. Enquanto isso, o algoritmo Merg Sort ou Quick Sort seria ineficiente. A classificação rápida é notoriamente desajeitada nos casos em que a lista está quase classificada. Obviamente, uma classificação por inserção nos objetos perturbados geralmente é melh

---

Exemplo de código em CeeLanguage

```
vazio
bubble_sort (int a[], int n /* o tamanho do array */)
{
	int eu;
	para (i = 0; i < n - 1; i++)
	{
		intj;
		para (j = n - 1; j > i; j--)
		{
			se (a[j - 1] > a[j])
			{
				/* Troca um par adjacente de itens. */
				int t = a[j - 1];
				a[j - 1] = a[j];
				a[j] = t;
			}
		}
	}
}
```

Essa formulação obtém o desempenho de melhor caso anunciado de O(n) . Caso contrário, melhor caso == pio caso == caso médio == O(n^2) .

```
vazio
bubble_sort (int a[], int n /* a parte do array que não está ordenada */)
{
	int i, classificado;
	do {
		classificado = 1;
		--n;
		para (i = 0; i < n; i++)
		{
			se (a[i] > a[i + 1])
			{
				/* Troca o par adjacente de itens. */
				int t = a[i];
				a[i] = a[i + 1];
				a[i + 1] = t;
				classificado = 0;
			}
```

```
                }
        } while (!classificado);
  }
```

Exemplo de código em PythonLanguage :

```
bolha def(alista):
        comprimento = len(alista)
        para i em xrange(comprimento-1):
                para j em xrange(comprimento-i-1):
                        if lista[j] > lista[j+1]:
                                lista[j], lista[j+1] = lista[j+1], lista[j]


 def flag_bolha(alista):
        comprimento = len(alista)
        para i em xrange(comprimento-1):
                trocado = Falso
                para j em xrange(comprimento-i-1):
                        if lista[j] > lista[j+1]:
                                lista[j], lista[j+1] = lista[j+1], lista[j]
                                trocado = Verdadeiro
                se não for trocado: retorno
```

```
 sort: aCollection
        ^aTamanho da coleção < self switchSize
        ifTrue: [^self insertionSort: aCollection]
        ifFalse: [^self quickSort: aCollection]
```

Eu testei uma classificação de bolha codificada rapidamente contra a classificação do Dolphin [DolphinSmallt ?] (que eu suponho ser QuickSort , como a maioria dos STs são, embora eu não tenha olhado.) Em 10 ou 20 elementos, a bolha foi mais rápida. Em 100, a classificação integrada foi cerca de 3x mais rápida. -- RonJeffrie.

Cuidado para encontrar o ponto de equilíbrio e, em seguida, repita, comparando o tipo de bolha com HeapSort

---

Eu sempre pensei que o Bubble Sort deveria ser chamado de Rock Sort porque os elementos em ordem realment caem para o fundo e não para o topo. Depende puramente se você corre para frente ou para trás.

```
 8 7 7 3
 7 8 3 5
 9 3 5 7
 3 5 8 8
 5 9 9 9
```

-- *CurtisCooley*

Veja também: BubbleSortChallenge

CategoriaClassificação de AlgoritmosAlgoritmos

Última edição 10 de dezembro de 2008