

Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG

Introsort

Augusto Oliveira, Gabrielle Coimbra, Matheus Mechetti

Introsort

Introsort ou introspective sort é um algoritmo de ordenação criado por David Musser em 1997. Ele começa com o quicksort e muda para o heapsort quando a profundidade da recursividade excede um nível baseado no (logaritmo do) número de elementos a ser classificados. É o melhor dos dois mundos, com um tempo de execução de pior caso de $O(n \log n)$ e desempenho prático comparável ao quicksort em conjuntos de dados típicos.

Ou seja: precisamos saber Heap e Quicksort.

Fonte: [Intro sort – Wikipédia, a enciclopédia livre](#)

Introsort

💡 Curiosidade: O Introsort foi criado para resolver o problema do pior caso do Quicksort ($O(n^2)$). Ele consegue manter a velocidade do quicksort mas com a segurança e garantia de desempenho do Heapsort.

Por combinar dois modelos, o paradigma desse algoritmo é do tipo **híbrido**, e seu método combina **particionar** e **selecionar**.

Fonte: [Intro sort – Wikipédia, a enciclopédia livre](#)

Complexidade

Quicksort

Melhor caso: $O(n \log n)$ ou $O(n)$

Pior caso: $O(n^2)$

Caso médio: $O(n \log n)$

Heapsort

Melhor caso: $O(n \log n)$

Pior caso: $O(n \log n)$

Caso médio: $O(n \log n)$

Complexidade

Introsort

Melhor caso: $O(n \log n)$

Pior caso: $O(n \log n)$

Caso médio: $O(n \log n)$

Adendo: Insertion Sort

Em alguns casos, o Introsort pode recorrer ao Insertion Sort. Este, por sua vez, tem como principal vantagem sua simplicidade e facilidade de lidar com pequenos conjuntos de dados.

Funcionamento

Para entender o Introsort, precisamos entender o pior caso do Quicksort

O pior caso será identificado pelo conceito de “profundidade de recursão”. Esse conceito diz respeito a quantas camadas estão empilhadas na memória.

Funcionamento

No pior caso do Quicksort (que pode ser causado por uma escolha ruim de pivô em um vetor quase ordenado, por exemplo), podem ocorrer estouros de pilha (stack overflow) e a lentidão de $O(n^2)$.

Assim, a regra proposta por David Musser é:

Limite = $2 \cdot \log_2 N$

// Limite é igual a duas vezes Log de n na base 2

Para 1000 elementos: $\log_2 1000 \approx 10$. Limite é aproximadamente 20.

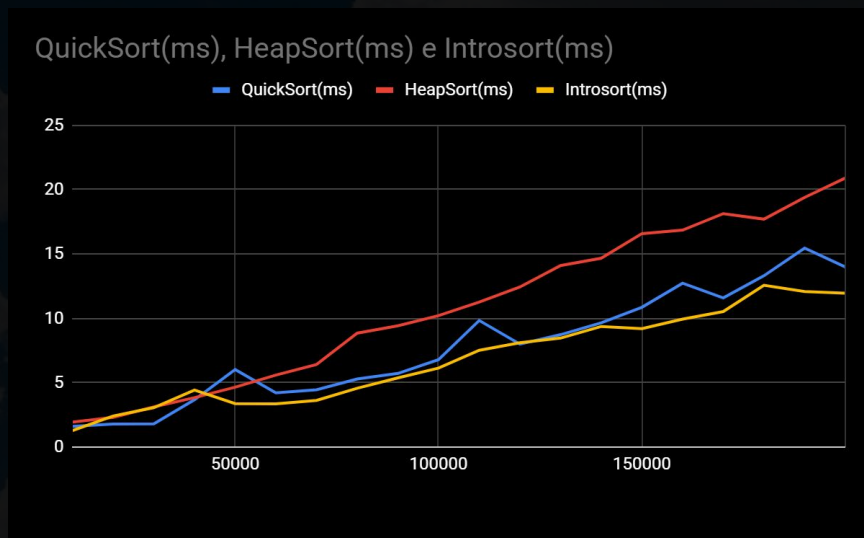
Funcionamento

O momento que o limite é atingido é quando o algoritmo muda seu método, passando a chamar o heapsort ao invés do quicksort.

💡 Curiosidade: Por que não o Mergesort? O Heapsort foi escolhido principalmente por não depender de memória extra.

Fonte: [IntroSort or Introspective sort - GeeksforGeeks](#)

Simulação



Acesse [esse link](#) para os dados completos da simulação. O código utilizado está disponível no GitHub, [clique aqui](#) para acessar.

Cálculo da Complexidade

Pior Caso: $O(n \log n)$.

Por que? Porque se o QuickSort tentar degradar para $O(n^2)$, o Introsort ativa o HeapSort, que tem um teto garantido de $O(n \log n)$.

Caso Médio: $O(n \log n)$.

Por que? Porque na maioria das vezes ele roda como um QuickSort, que é muito rápido na prática.

Complexidade de Espaço: $O(\log n)$.

Usa a pilha de recursão, assim como o QuickSort.

Vantagens e Desvantagens

Velocidade: Tão rápido quanto o QuickSort na maioria dos casos.

Segurança: Nunca "trava" ou demora uma eternidade (evita o pior caso $O(n^2)$).

Memória: É *in-place*, não precisa duplicar o array como o MergeSort.

O introsort está inserido na biblioteca padrão do C++ (`std::sort`)

Instabilidade: Assim como o QuickSort e o HeapSort, ele não é estável. Se você ordenar uma lista de "Pessoas" por idade, ele pode bagunçar a ordem alfabética original das pessoas com a mesma idade.

Complexidade de Implementação: É mais difícil de escrever do zero do que um algoritmo puro.

Finalizamos

