

Trabalho 1 - Análise de Algoritmos

Entrega: 25 de Setembro

Esse trabalho consiste em implementar o algoritmo de seleção em tempo linear visto em sala, e comparar com o algoritmo de seleção baseado em ordenação.

O trabalho deve ser realizado **em dupla**, e pode ser feito em qualquer linguagem.

O que tem que ser entregue **por email, em um zip com os nomes dos membros da dupla**:

1. Relatório de 2-3 páginas com o conteúdo pedido abaixo
2. Código fonte
3. Executável **windows**

Note que utilizarei sistema para controle de plágio de código, como o MOSS.¹

Tarefa 1: Seleção em tempo linear

Implemente o procedimento `LINEARSELECTION(A, k)` visto em sala que recebe uma lista A de números² e um inteiro k e retorna o k -ésimo menor elemento em A . Caso k esteja fora dos limites (i.e., ≤ 0 ou $> |A|$), retorne mensagem de erro/exceção. **A implementação tem que ser complexidade de tempo linear no pior-caso.**

Como base do código utilize os slides de aula, o Capítulo 9.3 do livro-texto “Introdução a Algoritmos” de Cormen, Leiserson, Rivest, e Stein, ou qualquer outra referência. Lembre de definir o caso-base da recursão.

No relatório você deve colocar:

1. Uma cópia da parte principal da função `LINEARSELECTION(A, k)` (deve ter uma estrutura semelhante à do pseudo-código dos slides de aula)
2. Indique **brevemente** porque o seu código satisfaz uma equação de recorrência da forma

$$T(n) \leq cst \cdot n + T(n/5) + T(7n/10)$$

(só precisa justificar a parte em azul).

3. Breve indicação de como rodar seu executável e formato da entrada esperado.

Tarefa 2: Experimentos

Realize experimentos comparando o `LINEARSELECTION(A, k)` implementado contra o algoritmo de seleção `SORTSELECTION(A, k)` baseado em ordenação (i.e., que computa a lista ordenada $A^{ord} \leftarrow$

¹<https://theory.stanford.edu/~aiken/moss>

²Pode fixar o tipo da entrada como lista/vetor de int/float/etc. caso ajude, não precisa fazer um código muito genérico.

`SELECT(A)` e retorna $A^{ord}[k]$. Você pode utilizar qualquer rotina de ordenação disponível (**menos CountingSort**), não precisa re-implementar.

Para os experimentos:

1. Sete sempre $k = \lfloor n/2 \rfloor$ (metade do tamanho da lista de entrada arredondada para baixo)
2. Para cada $n = 1.000, 2.000, \dots, 10.000$, gere 10 instâncias **aleatórias** de tamanho n (total de 100 instâncias). Os números em cada instância devem ser gerados aleatoriamente entre 1 e 100.000 (inteiros ou float/double).
3. Rode as algoritmos `LINEARSELECTION(A, k)` e `SORTSELECTION(A, k)` sobre essas instâncias. Para cada tamanho de instância n , compute a **média do tempo de execução** dos algoritmos (ou seja, compute 10 médias para cada algoritmo).

No relatório você deve reportar:

1. Um gráfico com o eixo x correspondendo ao tamanho da entrada n , e o eixo y correspondendo às médias dos tempos de execução dos algoritmos. Portanto o gráfico tem 2 séries, uma para cada algoritmo.
2. Compare **brevemente** esse gráfico com a complexidade teórica dos algoritmos.