

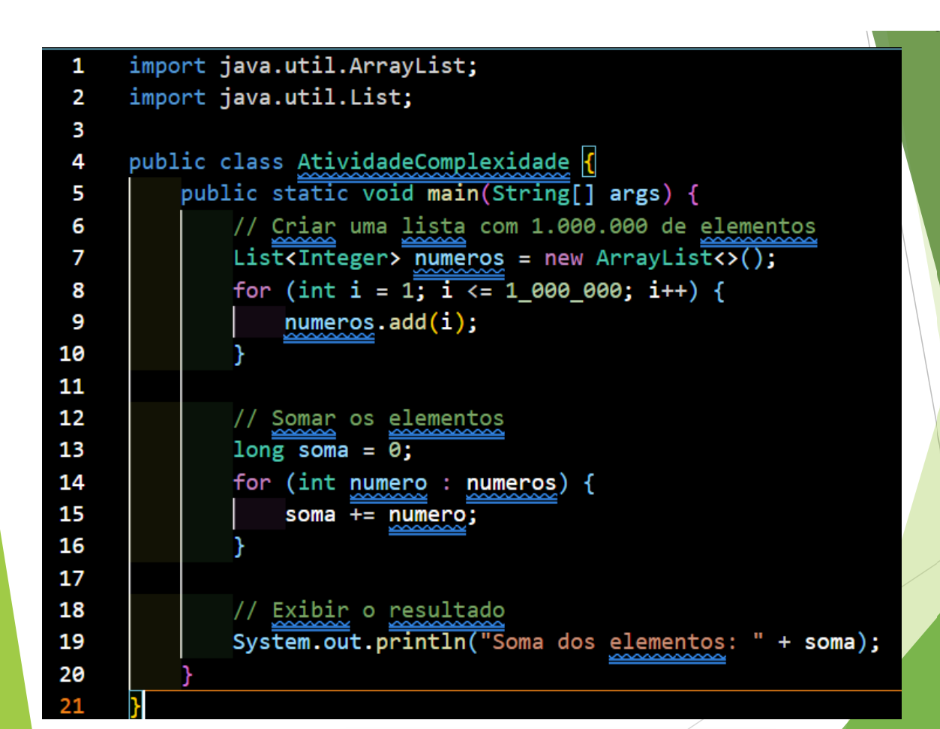
Curso Superior de Desenvolvimento de Software Multiplataforma

Nome: Lucas Vinicios Consani RA:3011392413046

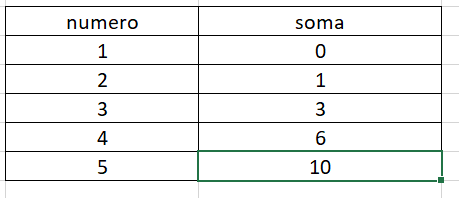
Matéria: Estrutura de dados

Prof helder Pestana

Exercico 1 – qual a complexidade do algoritmos e por que?



No primeiro exercício, o algoritmo está dividido em duas etapas principais: **criação de uma lista e soma dos elementos dessa lista.**

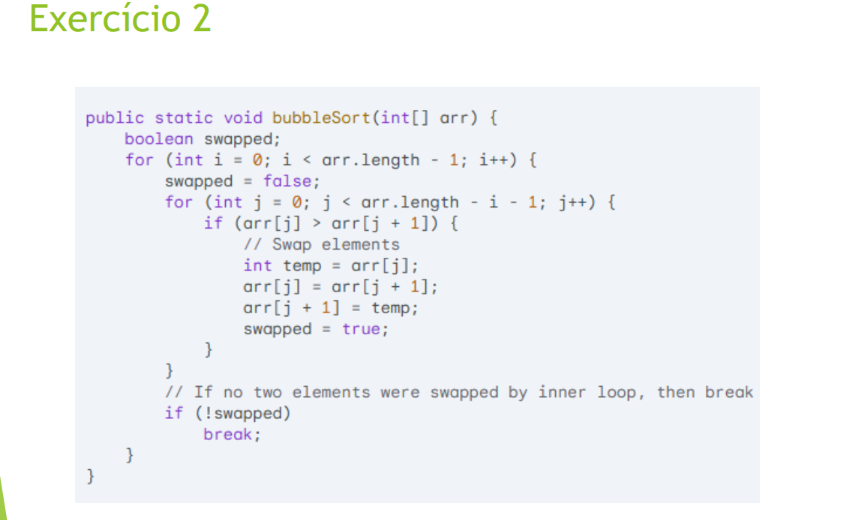


Sua funcionalidade seria aonde ele começa com um numero no caso 1 e a soma com o 0 a segunda etapa soma ambos e deixa salvo o resultado na lista e assim vai somando com o próximo numero exemplo 1 + 0 = 1 esse numero e salvo na lista e será somado com com o próximo numero que no caso e o 2 e assim continua 1 + 2 = 3 e esse numero sera salvo na lista e assim por diante ate 1000000.

Apesar de parecer simples, a implementação utiliza uma abordagem que demanda mais recursos do que o necessário. Além disso, a lista ocupa espaço na memória proporcional ao número de elementos, com uma complexidade de espaço de.

O problema principal desse código é o uso desnecessário de memória e processamento. A soma de números consecutivos de 1 até n pode ser calculada diretamente com a fórmula matemática, o que reduziria tanto a complexidade de tempo quanto a de espaço. Com isso, não seria necessário criar uma lista nem iterar sobre seus elementos, tornando a solução mais eficiente. Por isso, o código é considerado mais complexo do que o necessário, já que consome mais recursos computacionais para resolver um problema que possui uma solução muito mais simples e otimizada.

Exercício 2



Qual a complexidade de tempo no pior caso ?

O algoritmo Bubble Sort no pior caso, onde o array está ordenado em ordem reversa, terá uma complexidade de tempo de O(n^2). Isso porque o algoritmo precisa comparar e trocar cada par de elementos n vezes, o que leva a um tempo de execução proporcional ao quadrado do tamanho do array.

Como a otimização com a variável swapped influencia a complexidade?

O código implementa o algoritmo de ordenação "Bubble Sort" que ordena um array de inteiros. O algoritmo funciona percorrendo o array várias vezes, comparando elementos adjacentes e trocando-os se estiverem na ordem errada. O código inclui uma otimização que interrompe o processo de ordenação se nenhuma troca for realizada em uma iteração completa, indicando que o array já está ordenado. Em termos mais simples, o código organiza números em ordem crescente, comparando dois números ao lado e trocando-os se o primeiro for maior que o segundo. Ele continua fazendo isso até que todos os números estejam na ordem correta. A otimização faz com que o código pare de verificar se os números já estiverem em ordem, o que economiza tempo.

