UNIDADE PAULISTA 1

CIÊNCIA DA COMPUTAÇÃO

APS TÉCNICAS DE PROGRAMAÇÃO
-RECURSIVIDADE E ITERATIVIDADE-

Rafael Henrique Gonçalves Soares, 21566288

SÃO PAULO
2021

Recursividade X Iteratividade

O que é um programa com uma solução feita de maneira Recursiva ou Iterativa ? Ao se deparar com um código na qual é necessária uma solução com um determinado número de ações repetitivas, devemos analisar se o programa precisa do uso de sub-rotinas para a realização dessas ações, e é nesse momento que escolhemos usar funções (ou blocos de código) nomeadas de Recursivas ou Iterativas. Ambas essas funções tem a mesma finalidade, tratar a sub-rotina.

Uma função Iterativa é uma função na qual é realiza uma tarefa/processo na qual é repetido de novo e de novo durante o inicio do que chamamos de cada "iteração" anterior, a iteração é a própria realização da ação, sendo um processo rápido. Seu principal ponto negativo é a legebilidade de código, que para usuários mais experiêntes, conforme a tarefa, é necessário ter uma experiência maior na área para entender a finalidade e o método de execução da função

Entretanto a função Recursiva, embora tenha a mesma finalidade ela é basicamente um loop. Essa função tem a proposta de tornar a legebilidade do código mais simples e compacto em termos de uso de linhas de código. A sua principal característica é que ele é utilizado com um ponto de parada, na qual quando o loop finito da função acaba, a própria função acaba. Porém ela precisa de um maior uso da memória para guardar os retornos da função, tendo essa limitação de hardware, seu uso pode variar de máquina para máquina, também variando seu tempo de execução.

Qual escolher ? Depende do intuito do uso do programa e avaliação do usuário/cliente final, sobre a solução proposta. Como uma função recursiva, geralmente, tem limitações no número máximo de recursividades (seja em hardware, quanto ao tempo de processamento), uma plataforma que trabalha com um número descomunal/imenso de dados pode não terminar sua execução, devido à essas limitações, atrasando ou até mesmo parando o projeto/solução, porém para tarefas com poucos dados, seu uso para "a compactação" e entendimento do código, é uma opção bem vinda.

Já um função Iterativa pode ser utilizada para tratar um grande número de dados, mesmo que seu entendimento para usuários pouco experiêntes possa demorar, e mesmo que o código escrito fique maior.

TESTES DE FUNÇÕES

1. INTUITO DOS TESTES.

Utilizando a Linguagem Python, foi realizado pequenos testes “Caseiros” na plataforma Google Colab e no computador pessoal do aluno que está escrevendo esse trabalho (Rafael Soares). Nesses testes nos deparamos com o seguinte “problema” proposto pelo professor:

Há uma variável chamada “TAMANHO”, essa variável é responsável pela criação de uma lista/vetor nomeado originalmente de “colecao”, na qual a coleção será gerado uma lista de números entre 0 e 51, que posteriormente serão tratados nas funções nomeadas de “quickSort”, ao serem tratadas, esses dados serão dispostos na conclusão dos testes sobre velocidade, processamento e legebilidade de cada função quickSort

2. FUNÇÕES QUICKSORT

As funções quickSort criadas nesse programa, escritas pelo professor.André Santana (UAM, Campus Paulista 1, 2021- 2º Semestre. Turma de Ciência da Computação) são responsáveis pela Organização em ordem crescente dos números dispostos no nosso vetor/lista nomeada “colecao”, sendo uma função com uma soloção feita de forma Iterativa e outra Recursiva.

Ambas as funções recebem o parâmetro “partition”, que é um código feito para organizar a coleção, esse partition será repetido quantas vezes forem necessárias até a conclusão da organização, e quem irá ditar o momento de sua chamada será as funções de quickSort. Ou seja, ao final de tudo seria esquivalente a função “sort()” da biblioteca padrão do python, porém “na sua forma pura” já que foi escrita completamente do zero.

3. TESTES

Os testes consistem em registrar os tempos de execução de cada uma das funções, sua legebilidade e eficiência no consumo de recursos. Para isso foi utilizado a biblioteca Pandas e Matplotlib, para a criação de uma tabela excel e um gráfico demonstrativo com os resultados obtidos. Curiosamente os Resultados foram extremamente similares em ambas as funções, tanto pela execução no computador pessoal do aluno, quanto na execução através da plataforma “Google Colab”.

3.1.1. RESULTADOS NA MÁQUINA PESSOAL DO ALUNO

Antes de comentar sobre os resultados catalogados, será informado as especificações da máquina de testes do aluno, tratando-se de:

Processador: Core i5-9400

Ram: 8gb DDR4- 2666

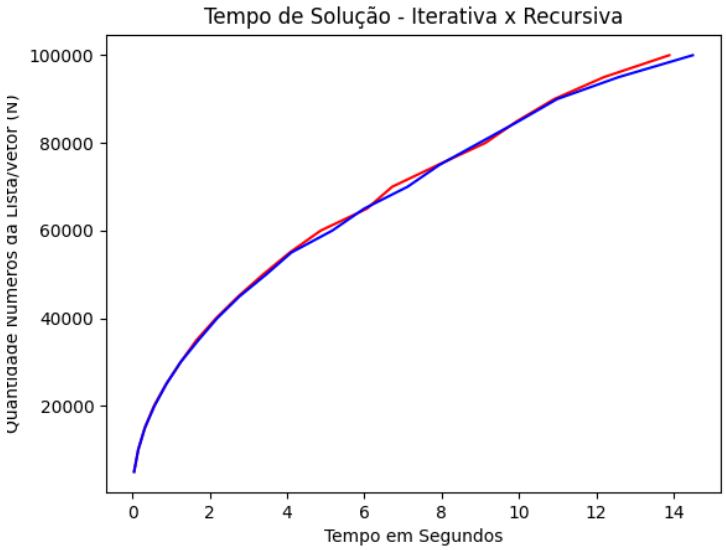
Armazenamento: SSD 128gb e 1(um) HD 1tb.

S.O: Windows 10

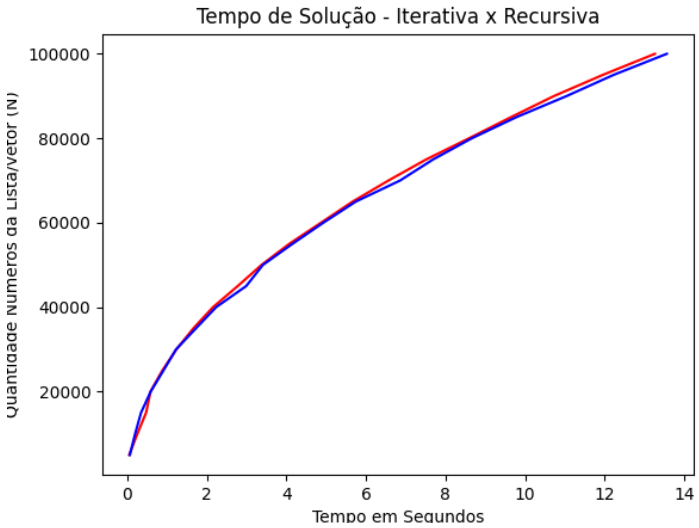
Para a realização dos testes o aluno realizou a criação de um Ambiente Virtual (virtual environment), que é uma unidade responsável pela instalação das bibliotecas utilizadas no código, além de não afetar nenhum outro projeto do aluno. Para os testes o aluno configurou esse ambiente com o Python 3.8.5 e instalou apenas as bibliotecas básicas do projeto (numpy, pandas, matlibplot, openpyxl).

OBS: Vale ressaltar que para o teste o aluno fez dois arquivos com o mesmo código, na qual um está com umas lista de tamanhos pré-configurada com 20 valores à serem utilizados na coleta de dados. E um arquivo para se usar até 20 valores/tamanhos inseridos pelo usuário (desde que esses valores não ultrapassem o limite de recursividade).

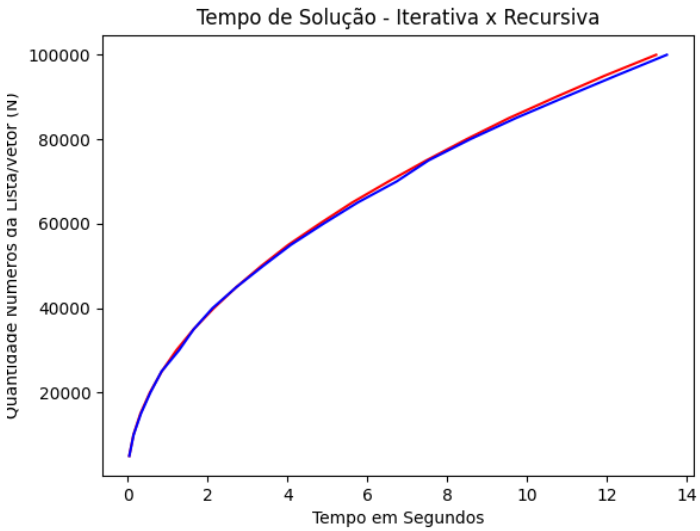
OBS 2: Os testes foram realizados 5x (vezes) na mesma máquina, nas mesmas condições, (com nenhum ou quase nenhum programa/tarefa auxiliar aberto, senão a IDE Visual Studio e o explorer do Windows)



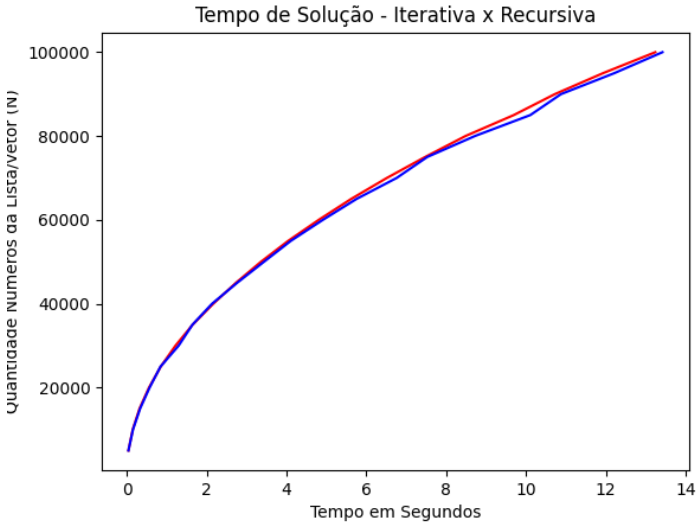
Teste 1



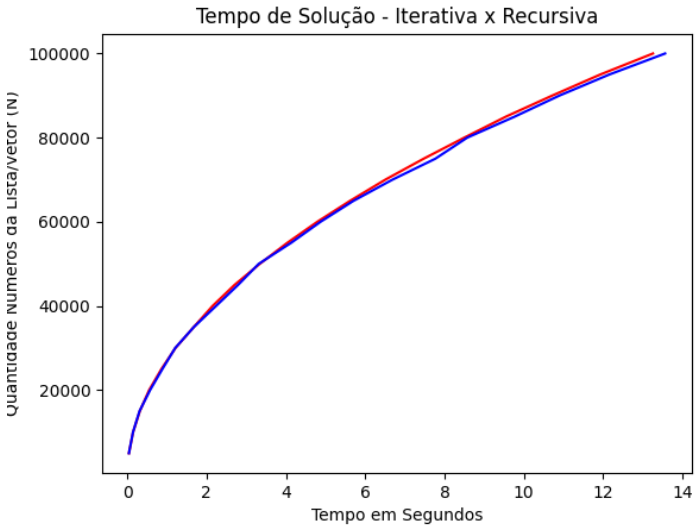
Teste 2



Teste 3



Teste 4



Teste 5

Os Resultados obtidos tem uma média de 4,81 segundos com recusrividade e 4,91 com iteratividade para a execução do programa e organizar os valores:

3.1.2. GOOGLE COLAB

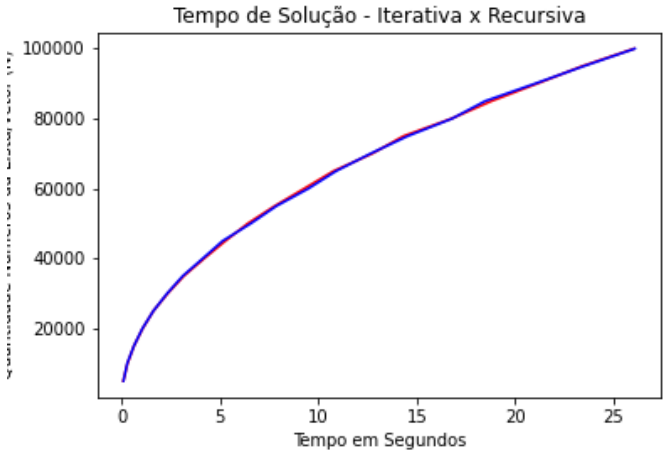
Com a ferramenta do Google Colab, ela foi realizada na seguinteconfiguração:

Google Compute Engine em Python 3

RAM: 12.69GB

Disco/Armazenamento: 107.72Gb

O gráfico gerado foi:



-Gráfico com plataforma do Google Colab .

Seus Resultados gerados em planilha Obtiveram um tempo de conclusão de 9,301s para a recursiva, e 9,307 na iterativa.

Porém vale lembrar que todos os valores foram bem próximos, e a solução recursiva nem sempre é a melhor, mesmo que a média com valores baixos, tenha sido quase idêntica.

Ao usar valores maiores, o tempo de espera de conclusão entre uma solução e outra ficou muito mais evidente, usando a própria plataforma do Google Colab. O teste foi feito chamando individualmente cada uma das funções, com a mesma quantidade de números gerados, porém ultrapassamos o limite de 100000 (cem mil) valroes para 5000000 (cinco milhões) de valores, na qual resultou nos seguintes tempos de conclusão (no terminal do Google Colab):

```
Tempo Solução Iterativa: 0.005833864212036133 s
Tempo Solução Iterativa: 708.3474171161652 s
```

708.3 Segundos com 5 Milhões de valores na iterativa

```
Tempo Solução Recursiva: 0.0054092407226562 5 s
Tempo Solução Recursiva: 837.7440056800842 s
```

837.7 Segundos na Recursiva

Ficando evidente claro, que a recursiva é mais demorada, quando se tratam grandes valores

4. CONCLUSÃO

Embora nos testes com valroes pequenos e com a máquina pessoal do aluno a Solução recursiva tenha apresentado bons resultados, com valores muito grandes, ela acabou por demorar muito mais tempo que a solução iterativa para ser finalizda, aproximadamente 2 minutos a mais.

Ambas as soluções são funcionais e boas de serem aplicadas, mas deve caber ao programador entender quais são os melhores cenários para se aplicar cada uma. Aplicando a iterativa em grandes tarefas que precisam ser feitas mais rpápidamente, ou a recursiva em tarefas “pequenas” que podem ser sujeitas à manutenção periódica e requerem um código mais “limpo” para serem feitas.