Universidade Federal do Rio Grande

Algoritmos e Estruturas de Dados II

Calculadora de Duas Operações

Bruno Agoston de Assis 126714

Rio Grande/2022

Introdução

Durante o desenvolvimento dos algoritmos foi desenvolvido duas versões, a primeira versão do algoritmo é o de adição, onde é feita a adição de uma sequência de números onde o usuário decide a quantidade de algarismos os operadores vão ter (fixo para os dois operadores) e depois decide qual algarismo será o algarismo sequencial. Abaixo segue o pseudocódigo do algoritmo de adição

Pseudocódigo do algoritmo de adição:

```
#Pseudo Codigo

#Adição


sum_operation(size_of_operators):

1.  vectors vector_operator1[size_of_operators], vector_operator2[size_of_operators], vector_result[size_of_operators+1];
2.  int aux_sum, aux_result;
3.  string operator_of_sequence1 , operator_of_sequence2;
3.  read(operator_of_sequence1);
4.  read(operator_of_sequence2);
5.  vector_operator1 <- fill_vector(size_of_operators, operator_of_sequence1);
6.  vector_operator2 <- fill_vector(size_of_operators,operator_of_sequence2);
7.  for i<-size_of_operators until 0 do:
8.        aux_result <- vector_operator1[i] + vector_operator2[i] + aux_sum;
9.        vector_result[i+1] <- aux_result mod 10;
10.       aux_sum <- aux_sum div 10;
11. end for;
12. vector_result[0] <- aux_div;
13. show(vector_result);
14. end
```

O segundo algoritmo que é o de multiplicação , utiliza a multiplicação no modelo americano e tem o mesmo funcionamento, inicialmente se lê o tamanho dos operadores e depois qual será o algarismo sequencial.

Pseudocódigo do algoritmo de multiplicação:

```
multiplicationPseudocode(size_of_operators)

1.  vector vector_operator1[size_of_operators], vector_operator2[size_of_operators], vector_result[size_of_operators*2];
2.  int aux_multiplication, aux_sum, aux_result, vector_result_aux[size_of_operators+1];
3.  string operator_of_sequence1 , operator_of_sequence2;
3.  read(operator_of_sequence1);
4.  read(operator_of_sequence2);
5.  vector_operator1 <- fill_vector(size_of_operators, operator_of_sequence1);
6.  vector_operator2 <- fill_vector(size_of_operators,operator_of_sequence2);
7.  for i <- 0 until (size_of_operators*2) do:
8.      vector_result[i] <- 0
9.  for i <- size_of_operators until 0 do:
10.         aux_multiplication <- 0
11.         for j <- size_of_operators until 0 do:
12.             aux_result <- vector_operator1[j] * vector_operator2[i] + aux_multiplication
13.             vector_result_aux[j+1] <- aux_result mod 10
14.             aux_multiplication <- aux_result div 10
15.         end for;
16.         vector_result_aux[0] <- aux_multiplication
17.         aux_multiplication <- 0;
18.         for j <- size_of_operators until 0 do:
19.             aux_sum <- vector_result_aux[j+1] + vector_result[i+j] + aux_multiplication
20.             vector_result[i+j] <- aux_sum mod 10;
21.             aux_multiplication <- aux_sum div 10;
22.         end for;
23.     end for;
24.     show(vector_result)
end;
```

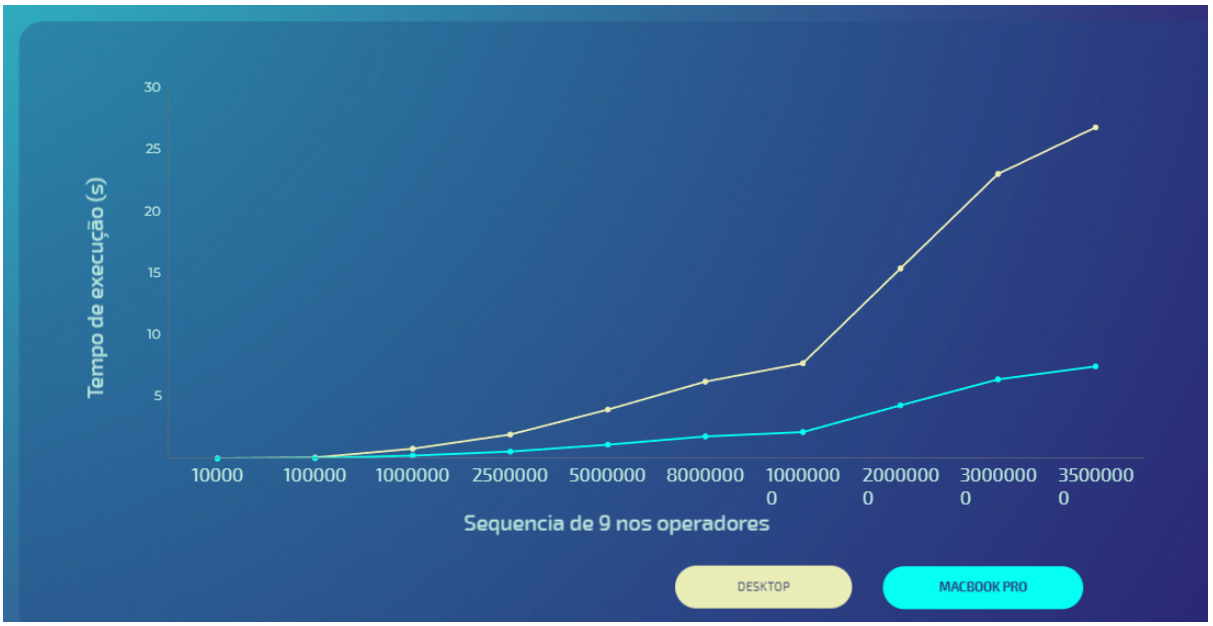
Especificação dos computadores

Foi escolhido um computador desktop e um notebook para a execução dos algoritmos, a configuração deles são:

	Configuração Maquinas	
	Desktop	**Notebook (Macbook Pro 16')**
OS:	Windows 10 Pro 21H2	macOS Monterey 12.3.1
Processador	AMD Ryzen 5 2600 6-Core 3.40 GHz	Intel Core i7 6-Core 2,6Ghz
RAM	16 GB 2666Mhz DDR4	16 GB 2667MHz DDR4
Placa de Video	Radeon RX 580 OC 8GB	AMD Radeon Pro 5300M 4GB
Disco Rigido	SSD NVMe 512GB WDBlack	SSD NVMe 512GB Apple

Gráficos

Gráfico de tempo de execução no algoritmo de adição



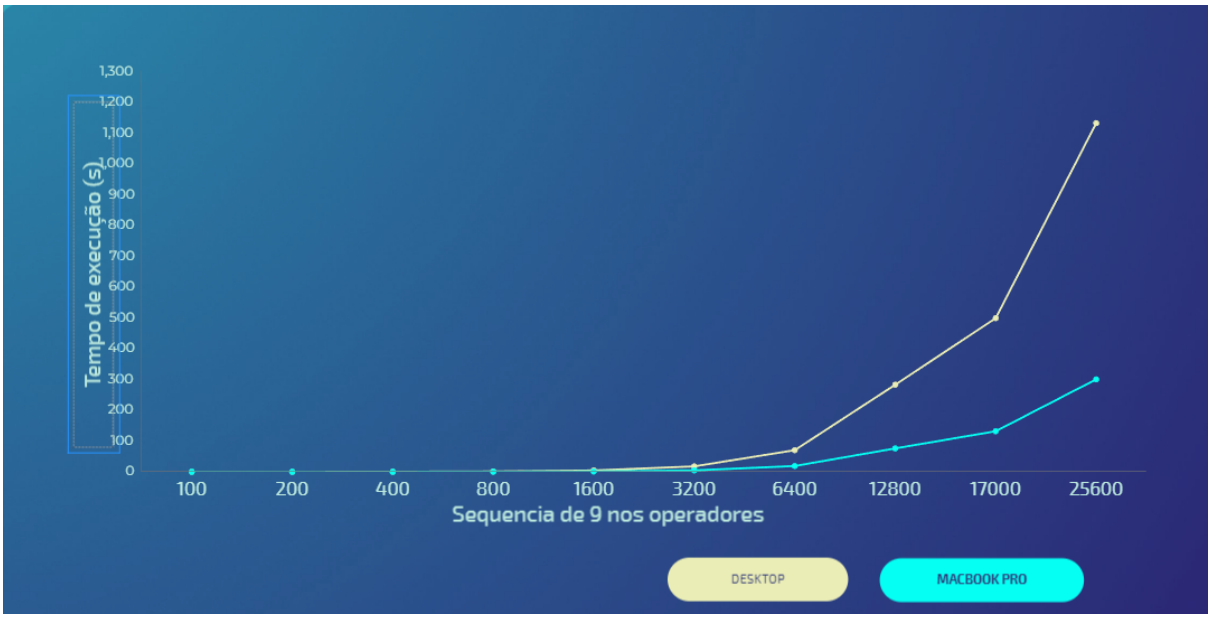
Valores

Adição			
Desktop		Notebook Apple	
Sequencias de algarismos 9	Tempo de execução (s)	Sequencias de algarismos 9	Tempo de execução (s)
10000	0.008001327514648438	10000	0.002663135528564453
100000	0.0780181884765625	100000	0.0269010066986084
1000000	0.7891800403594971	1000000	0.23131394386291504
2500000	1.9403460025787354	2500000	0.5605731010437012
5000000	3.959404468536377	5000000	1.115530014038086
8000000	6.226407051086426	8000000	1.7851531505584717
10000000	7.7153449058532715	10000000	2.138819932937622
20000000	15.397742509841919	20000000	4.303348064422607
30000000	23.055995225906372	30000000	6.409424781799316
35000000	26.82895255088806	35000000	7.460489988327026

Análise dos resultados

Podemos verificar que há um crescimento linear no tempo de processamento nos dois casos. Porém comparando os tempos de execução do Desktop com o Notebook vemos que em todas as comparações o Notebook ganha em tempo de execução tanto nas sequências menores , quanto nas sequências mais longas.

Gráfico de tempo de execução no algoritmo de Multiplicação



Valores

Multiplicação			
Desktop		Notebook Apple	
Sequencias de algarismos 9	Tempo de execução (s)	Sequencias de algarismos 9	Tempo de execução (s)
100	0.01600337028503418	100	0.004748106002807617
200	0.07101631164550781	200	0.01922893524169922
400	0.27506160736083984	400	0.07963705062866211
800	1.1302568912506104	800	0.30522799491882324
1600	4.428040981292725	1600	1.1992499828338623
3200	17.72210144996643	3200	4.707338094711304
6400	70.1795301437378	6400	18.837880849838257
12800	282.53581976890564	12800	75.92170190811157
17000	498.57708048820496	17000	131.6718327999115
25600	1131.444151878357	25600	300.06819891929626

Análise dos resultados

Podemos analisar que assim como no gráfico demonstrado de adição, há uma diferença notável nos tempos de execução por quantidade de algarismo que temos entre o Desktop e o notebook. Nos valores finais , vemos uma diferença significativa de 3x tempo de execução no desktop , comparado ao Macbook.

Conclusões finais

Vemos que notavelmente o Macbook Pro da Apple tem um desempenho extremamente superior comparado ao Desktop. Mesmo que os hardware sejam próximos, vemos que a diferença das execução acaba não sendo.
Acredito que uma das diferenças para causar tanta diferença no tempo de execução, se dá pelo sistema operacional utilizado. Enquanto o notebook utiliza o macOS da apple, o desktop utiliza o Windows 10 Pro, o que pode haver diferenças na forma que os códigos são interpretados para execução.