**Execute o comando time ./exercicio 1000 200 > saida.txt (como a saída do programa é grande, vamos redirecioná-la para o arquivo saída) e veja quanto tempo o programa gasta para ser executado. O comando time do linux pode ser utilizado para medir o tempo de execução de um programa: o tempo “real” indica o tempo “medido no relógio” (também conhecido por “wall time”); o tempo “user” indica o tempo de uso da CPU e o tempo “sys” indica o tempo utilizado com chamadas ao sistema operacional.**

**Antes:**

real 0m0,006s

user 0m0,000s

sys 0m0,006s

**Depois:**

real 0m0,005s

user 0m0,005s

sys 0m0,000s

**Flush:**

real 0m0,004s

user 0m0,003s

sys 0m0,000s

**-O3:**

real 0m0,004s

user 0m0,001s

sys 0m0,004s

**Execute novamente o comando passando como parâmetro os valores 10000 e 10000**

**Antes:**

real 0m0,250s

user 0m0,237s

sys 0m0,012s

**Depois:**

real 0m0,021s

user 0m0,013s

sys 0m0,009s

**Flush:**

real 0m0,007s

user 0m0,004s

sys 0m0,003s

**-O3:**

real 0m0,006s

user 0m0,006s

sys 0m0,000s

**Execute novamente o comando passando como parâmetro os valores 10000 e 20000.**

**Antes:**

real 0m0,492s

user 0m0,476s

sys 0m0,016s

**Depois:**

real 0m0,037s

user 0m0,017s

sys 0m0,020s

**Flush:**

real 0m0,007s

user 0m0,007s

sys 0m0,000s

**-O3:**

real 0m0,008s

user 0m0,008s

sys 0m0,000s

**Execute novamente o comando passando como parâmetro os valores 20000 e 10000.**

**Antes:**

real 0m0,479s

user 0m0,463s

sys 0m0,016s

**Depois:**

real 0m0,020s

user 0m0,005s

sys 0m0,015s

**Flush:**

real 0m0,006s

user 0m0,006s

sys 0m0,000s

**-O3:**

real 0m0,006s

user 0m0,006s

sys 0m0,000s

**Execute novamente o comando passando como parâmetro os valores 30000 e 30000.**

**Antes:**

real 0m2,087s

user 0m2,051s

sys 0m0,036s

**Depois:**

real 0m0,048s

user 0m0,012s

sys 0m0,036s

**Flush:**

real 0m0,009s

user 0m0,005s

sys 0m0,004s

**-O3:**

real 0m0,009s

user 0m0,009s

sys 0m0,000s

**Execute novamente o comando passando como parâmetro os valores 100000 e 30000.**

**Antes:**

real 0m6,931s

user 0m6,902s

sys 0m0,028s

**Depois:**

real 0m0,048s

user 0m0,016s

sys 0m0,032s

**Flush:**

real 0m0,010s

user 0m0,009s

sys 0m0,000s

**-O3:**

real 0m0,007s

user 0m0,007s

sys 0m0,000s

**Faça outros testes.**

**30000 e 100000:**

**Antes:**

real 0m6,942s

user 0m6,822s

sys 0m0,120s

**Depois:**

real 0m0,142s

user 0m0,049s

sys 0m0,093s

**Flush:**

real 0m0,018s

user 0m0,018s

sys 0m0,000s

**-O3:**

real 0m0,018s

user 0m0,018s

sys 0m0,000s

**40000 e 40000:**

**Antes:**

real 0m3,688s

user 0m3,627s

sys 0m0,060s

**Depois:**

real 0m0,061s

user 0m0,021s

sys 0m0,041s

**Flush:**

real 0m0,010s

user 0m0,010s

sys 0m0,000s

**-O3:**

real 0m0,010s

user 0m0,005s

sys 0m0,005s

**50000 e 50000:**

**Antes:**

real 0m5,734s

user 0m5,681s

sys 0m0,052s

**Depois:**

real 0m0,075s

user 0m0,044s

sys 0m0,032s

**Flush:**

real 0m0,013s

user 0m0,013s

sys 0m0,000s

**-O3:**

real 0m0,012s

user 0m0,012s

sys 0m0,000s

**Pergunta: com base nos resultados obtidos nos testes, como você acha que os parâmetros do programa influenciam no tempo de processamento?**

Quanto maior os parâmetros passados, maior será o tempo de processamento. Além disso, os dois parâmetros influenciam de maneira similar no tempo total de processamento pois a parte do código que mais pesa no processamento é a condição *“vetor[j] == i”* pois ela é executada *“limiteRand \* numTestes”* vezes.

**Analisando o código, qual a parte do programa você acha que “pesa” mais no tempo de processamento?**

A parte do código que mais pesa no processamento é a condição *“vetor[j] == i”* pois ela é executada mais vezes que todas as demais *(limiteRand \* numTestes)*.

**Modifique o programa de forma que ele fique (BEM) mais eficiente (dica: o código pode ser muito melhorado modificando-se a ideia do algoritmo).**

**Após melhorar o programa, troque o endl por um “\n” e, antes do return 0;, coloque um cout << flush. Veja a influência que isso tem no tempo de processamento. Explique tal influência.**

O que o *“cout << flush”* faz é que ao invés de imprimir tudo no arquivo *“saida.txt”* de uma vez no final, ele vai imprimindo a cada iteração.

**Finalmente, ao compilar a última versão do programa, adicione a flag “-O3” às diretivas do g++. Teste o programa novamente e explique a diferença de tempo.**

O que a flag *“-O3”* faz é usar vários algoritmos para analisar o código, a fim de converter para o menor e mais rápido código de máquina, com o custo de uma compilação mais longa.