Universidade de São Paulo - USP
Bacharelado em Ciência da Computação
Disciplina: Técnicas de Programação I (MAC0216)
Professor: Daniel Macêdo Batista
Acadêmico: Renan Ryu Kajihara
Número USP: 14605762

Tabela 1: Desempenho das funções em termos do tempo de execução em milissegundos

Função	Média	Mínimo	Máximo
ep1Passo1Preenche	70,9	38	113
ep1Passo2XOR	172,7	81	254
ep1Passo3Comprime	182,7	86	280
ep1Passo4Hash	0,5	0	1
ep1Passo4HashEmHexa	0,9	0	1
ep3CriaVetorMagico	1021,5	681	1304
ep3CalculaEntropiaShannon	80,4	52	103

Conclusão

Primeiramente, é essencial ressaltar o fato de que as Strings utilizadas para a medição do tempo de todas as funções, em exceto a função ep3CriaVetorMagico, possuíam apenas 10 caracteres. Dessa forma,o tempo de execução foi muito baixo, podendo ser muito maior caso as Strings possuam um tamanho maior. Além disso, evidencia-se que, pelo fato da linguagem de programação C possuir um baixo nível de abstração e ser compilada, a execução das funções é bem mais rápida do que se fossem executadas em outra linguagem, como Python, por exemplo.

Por meio da tabela, é possível observar que a função “ep3CriaVetorMagico” foi a que mais demorou para ser executada. Tal fato ocorre pois, uma vez que, para gerar um vetor de 256 números, sem repetição, a função “rand()” deverá gerar muitos números, já que, se “rand()%256” já estiver no vetor, a função deverá gerar um novo número. Além disso, para verificar se um número já está no vetor, é necessário percorrer todo o vetor já gerado, sendo tal processo realizado toda vez em que é gerado um novo número.

Ademais, é notório que as funções “ep1Passo4Hash” e “ep1Passo4HashEmHexa” são as que têm menor tempo de execução. Isso ocorre porque as funções são muito simples e trabalham com vetores com poucos elementos.

As funções “ep1Passo2XOR” e “ep1Passo3Comprime” foram as funções, que envolviam a criação do hash, que mais demoraram para serem executadas. Isso decorre do fato de que ambas as funções possuem vários laços, que percorrem uma quantidade significativa do vetor de entrada. Nesse sentido, tais funções demorariam muito mais tempo se as Strings de entrada possuíssem mais caracteres.

A função “ep1Passo1Preenche” demorou muito menos que as funções “ep1Passo2XOR” e “ep1Passo3Comprime”, pois essa função apenas copia a string de entrada e preenche as demais posições para que o vetor seja múltiplo de 16. Dessa forma, ela possui apenas um laço que percorre o vetor de entrada, justificando assim, a sua rapidez comparada às outras duas.

Por último, a função “ep3CalculaEntropiaShannon” foi bastante rápida apesar de possuir uma quantidade considerável de laços. Tal fato ocorre, pois as Strings de entrada possuíam poucos caracteres, fazendo com que o acesso ao vetor não fosse executado diversas vezes.

Configuração do Computador

Todas as execuções ocorreram em um computador com processador Intel Core i7, com memória de 15,4 GB e com Sistema Operacional Ubuntu na versão 20.04.4 LTS.