Universidade Tecnológica Federal do Paraná
Disciplina: Algoritmos e Estruturas de Dados 2
Aluno: Jean Carlos Martins Miguel RA 1640593
Professor: Dr. Juliano Henrique Foleis

Semana 4: Ordenação – Counting Sort e Radix Sort

2. Compare o tempo de execução do Radix Sort na base 2 e do Radix Sort na base 10 ordenando vetores aleatórios gerados usando int* random_vector(int n, int max, int seed) com n = 1000, 10000, 100000, 500000, 1000000, max = n * 100 e seed = 0. Anote os resultados na Tabela abaixo em milisegundos (ms).

	1000	10000	100000	500000	1000000
RadixSort (base 2)	0.820000	8.979000	43.170000	114.606000	225.765000
RadixSort (base 10)	1.114000	7.036000	36.211000	91.852000	182.139000

3. Analisando os resultados obtidos, qual versão foi mais rápida? Por quê?
R: A versão mais rápida foi a do RadixSort (base 10), por pouca diferença, apesar da função do radix base 10 ao usar o counting sort ter que fazer o cálculo utilizando duas operações (divisão e resto) ainda é mais rápido que a função do radixSort base 2 não por causa da operação em si, mas pelo fato de que para representar um número em decimal tem-se menos dígitos do que em binário, por exemplo: o número 10 em base decimal tem 2 dígitos , já em binário ,1010 tem 4 dígitos fazendo com que o counting sort seja executado mais vezes nesse caso devido a maior quantidade de dígitos não pela operação de deslocamento ,mas pela quantidade de dígitos para representar um número em binário, isso que faz com que o RadixSort na base 10 seja um pouco mais rápido,por ter menos dígitos na representação decimal.