CSE 7350

Course Timeslot and Student Assignment Project

Richard Simon

Computer Science
Southern Methodist University
United States
22 October, 2019

Contents

1 Computing Environment

The computing environment I used for testing is a desktop workstation. The machine is running Arch Linux, kernel version 5.3.7-arch1-1-ARCH. The machine has an Intel i9 9900k 16 core CPU pegged at 5.0GHz.

2 Data Generation

2.1 Uniform Distribution

2.1.1 Method

This data generator will produce a distribution such that each course number always has an equal probability of being selected next. The algorithm I chose to implement this functionality is aptly named a uniform shuffle. The algorithm is quite straightforward. The algorithm operates on each element $a_i \mid a \in [a_i...a_0]$, choosing a random element $a_j \mid j >= i$ and swapping the elements.

2.1.2 Runtime

Because the algorithm operates on each element exactly once, the runtime is tightly bounded by a linear function. That is, the function expressing the runtime of the algorithm is $\Theta(n)$, where n is the difference between the maximum output value of the generator, and the minimum (the total number of elements in the buffer that will be operated on). Each time the full buffer of shuffled courses has been yielded, the generator will re-shuffle the buffer to ensure uniformity. Because of this, when generating the course numbers, the runtime will be roughly $\Theta(C * S * K)$. The runtime data for uniform data generation bellow support a runtime function $\Theta(C * S * K)$. The number of possible courses was held constant for this experiment, to better illustrate the linear relationship S and K have with the total runtime.

c = 1,000	Courses / Student	100	200	400	800	1600	3200
# Students							
100		0.005s	0.006	0.008	0.012	0.20	0.038
200		0.006	0.008	0.012	0.021	0.038	0.072
400		0.008	0.013	0.021	0.036	0.072	0.142

2.2 Skewed (Linear) Distribution

2.2.1 Method

The method to create a linear distribution of data is rather straightforward also. I drafted off of my efforts for the uniform distribution to create a skewed one. Since each number generated from a uniform number generator has an equal probability of being chosen, creating 2 generators with different random seeds, and taking a max of their yields will create a roughly linear distribution, skewed towards larger course numbers.

2.2.2 Runtime

Because each generator has a linear time complexity to properly shuffle its data, running 2 of them serially will have a time complexity of $\Theta(2n)$