Illinois Institute of Technology

Department of Computer Science

CS430 - Section: 04 (live)

Introduction to Algorithms

IIT Assignment 2 - Report

Authors:
Bastien LEDUC
Ashik SYED SHAFFIULLAH
Kachikwu NWIKE
Hongbo WANG

Student Number:
A20520860
A20516459
A20517226
A20524770

November 4, 2022

Instructions

1. We would like to generate samples of a random variable X uniformly distributed over the interval (a, b), $0 < a < b$.

   (a) **(2 points)** What is the probability density function $f(x)$ of X? Find the corresponding cumulative distribution function (CDF) $F(x)$ of X.

   (b) **(2 points)** Use the inverse transform method to generate X given a generator of a random variable U uniformly distributed over the interval $(0, 1)$. Present a **formal mathematical justification** that your algorithm for generating X is correct.

   (c) **(4 points)** Generate three sequences $X_i = x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,n}$, where $i = 1, 2, 3$, of numbers ($n = 10000$ samples each) uniformly distributed over the intervals (a_i, b_i). Plot their **empirical CDFs** $F_i^{(e)}(x_j)$, where $j = 1, 2, 3, \ldots, k$. Assume $k << n$. What **sorting algorithm** did you use to calculate numerical values of $F_i^{(e)}(x_j)$? Find the sorting algorithm **minimizing** the complexity $T(n)$.

   (d) **(1 points)** Compare your empirical $F_i^{(e)}(x_j)$ and theoretical $F_i(x_j)$ CDFs.

   Submit source codes of all the programs you need to produce your results.

Solutions

1. (a) The probability density function $f(x)$ of a random variable X uniformly distributed over the interval (a, b), $0 < a < b$ is:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a, b] \\ 0 & \text{if } x \notin [a, b] \end{cases} = \frac{1}{b-a} \mathbb{1}_{[a,b]}(x)$$

The corresponding CDF of X is denoted by:

$$F_X(x) = P(X \leqslant x) = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } a \leqslant x \leqslant b = U([a, b]) \\ 1 & \text{if } x > b \end{cases}$$

(b) The idea of the inverse transform method is to generate a random number from any probability distribution by using its inverse CDF as follows (for a continuous r.v.):

$$X = F_X^{-1}(U)$$

Here, X has a distribution function F because, as U is a Uniform random variable on $(0, 1)$, we know that $P(U \leqslant y) = y$, for $0 \leqslant y \leqslant 1$. Thus:

$$\begin{aligned} P(X \leqslant x) &= P(F^{-1}(U) \leqslant x) \\ &= P(F(F^{-1}(U)) \leqslant F(x)) \\ &= P(U \leqslant F(x)) = F(x) \end{aligned}$$

as required. The key step is that $F(F^{-1}(U)) = u$ for each u, which is clearly true when F is strictly increasing.

In our case, we denote $F_X(U)^{-1}$ by:

$$X = F_X^{-1}(U) = U \times (b - a) + a$$

where U is a vector of uniform probabilities and a and b the range values of X. It is used to generate a random variable X over an interval (a, b) (where $0 < a < b$) when given as input a random variable U uniformly distributed over the interval $(0, 1)$, which can be interpreted as probabilities. Therefore, by isolating the X from the CDF expression defined in 1.(a), we can find the value of X between any interval (a, b) given its cumulative probability U. We can see in the figure 1 the distribution comparison between the generated and actual uniform r.v., where they both tend to the shape of the PDF function as the number of sample increases.

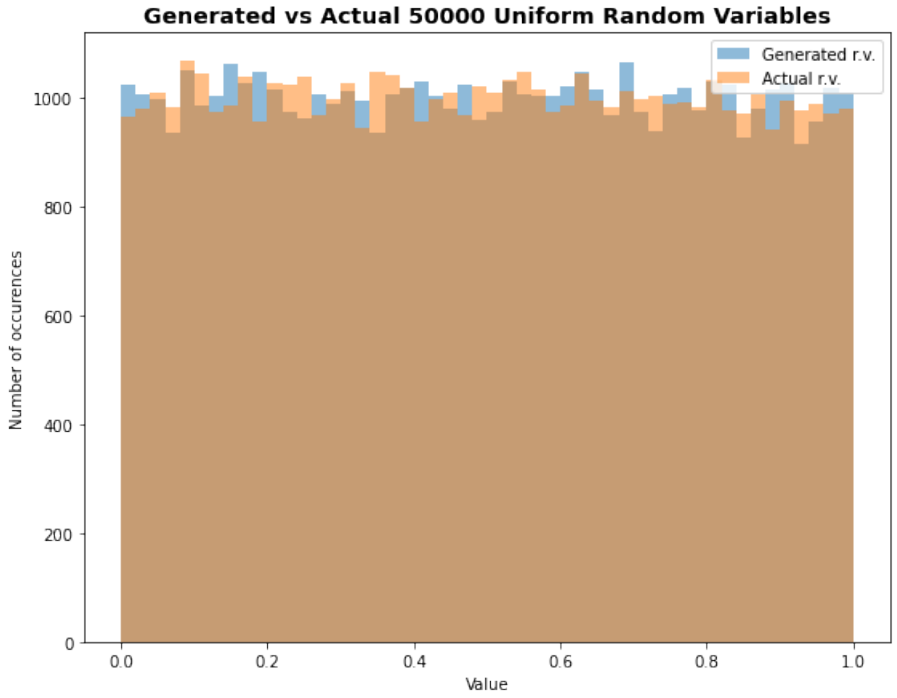

Figure 1: Generated vs Actual Uniform Random Variables.

(c) For each sequence X_i, we first generate $n = 10000$ samples of uniformly distributed values between $(0, 1)$, which we are interpreting as CDF probabilities. Then, we generate our X_i sequence between (a_i, b_i) using the inverse transform method of the uniform distribution. After that, we need to plot the empirical cumulative distribution function (ECDF) of our generated X_i. The ECDF is calculated by ordering all of the unique observations in the data sample and calculating the cumulative probability for each as the number of observations less than or equal to a given observation divided by the total number of observations. It can be expressed like so:

$$\hat{F}(x_j) = F_i^{(e)}(x_j) = \frac{\#\ of\ observations \leqslant x_j}{k}$$

where $j = 1, 2, 3, \ldots, k$ and $k << n$ is the size of the sorted data sample made from the first k accessed values of the sequence X_i. Therefore by definition, we need to sort those k values before computing the ECDF.

To determine the best sorting algorithm, we ran experiments on the different algorithms discussed in class, which includes insertion sort, merge sort, heap sort, quick sort and bucket sort. We ran each sorting algorithm 10 times and calculated the average time it took them to sort arrays of sizes from 10 to $k = 1000 << n$. After comparing the time of each algorithm, we found out that the bucket sort algorithm is the one that minimizes the complexity $T(n)$. Since the sequences are uniformly distributed over (a_i, b_i), the bucket size will almost be the same for all the buckets.

$O(n)$ is the complexity to store all elements in the bucket list where n is the number of elements. $O(k)$ is the complexity for sorting each bucket where k is the number of buckets. Thus, $T(n) = O(n + k)$ for the bucket sort which is the optimal sorting algorithm for minimizing $T(n)$ as it is linear (see figure 2 below).

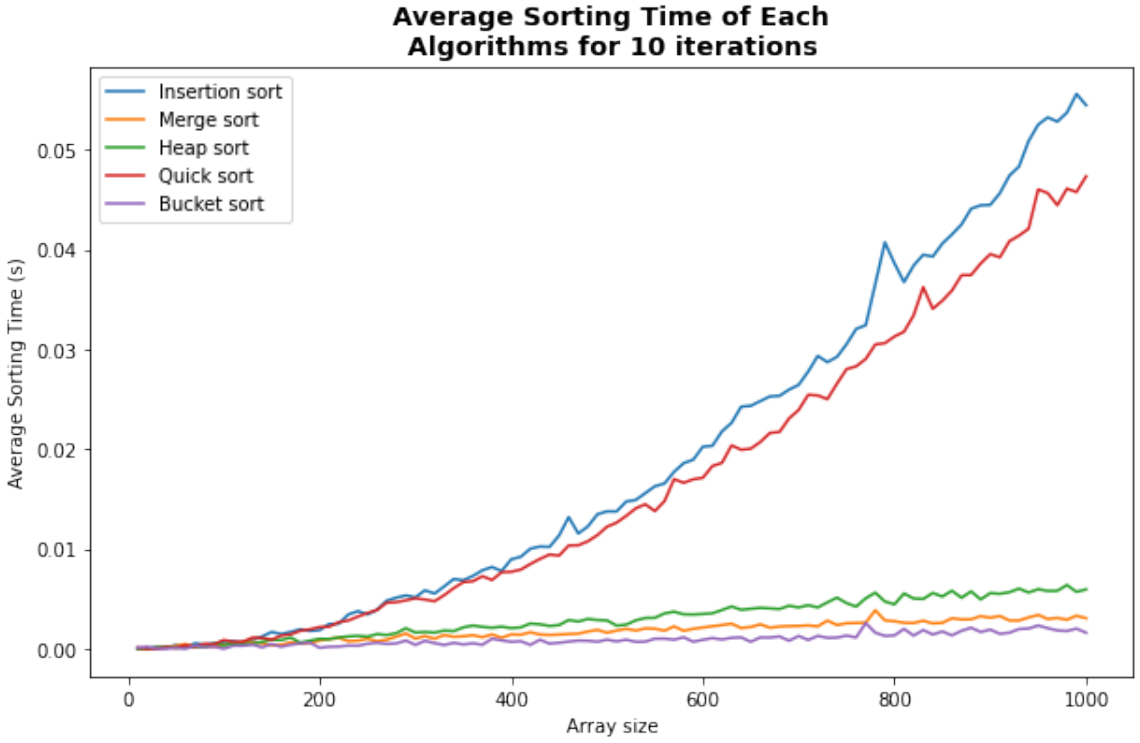

Figure 2: Time complexity comparison of different sorting algorithms.

(d) The empirical CDF usually approximates the CDF quite well, especially for large values of k. We can observe in the Figures 3, 4, and 5 as k increases, the ECDF (in blue) seems to getting closer to the linear CDF of the uniform distribution (in orange). Therefore, the generated X_i values seem to be well represented by the uniform distribution when the k value is high enough from the assumption $k << n$.

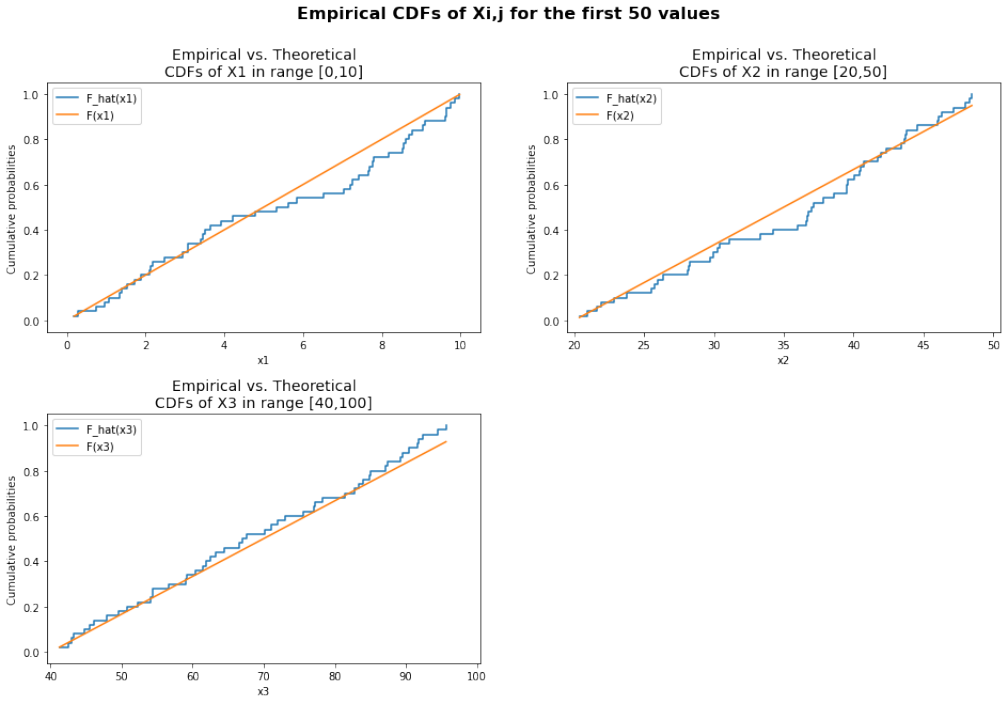

Figure 3: CDF and ECDF for the first 50 values of the unsorted generated X_i sequences.

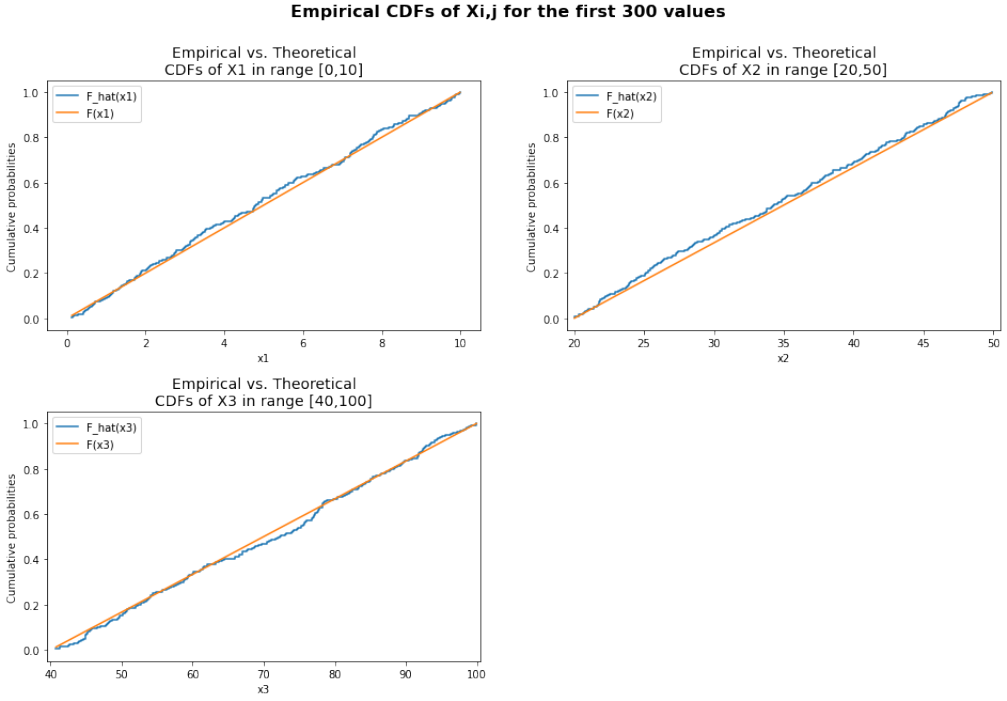

Figure 4: CDF and ECDF for the first 300 values of the unsorted generated X_i sequences.

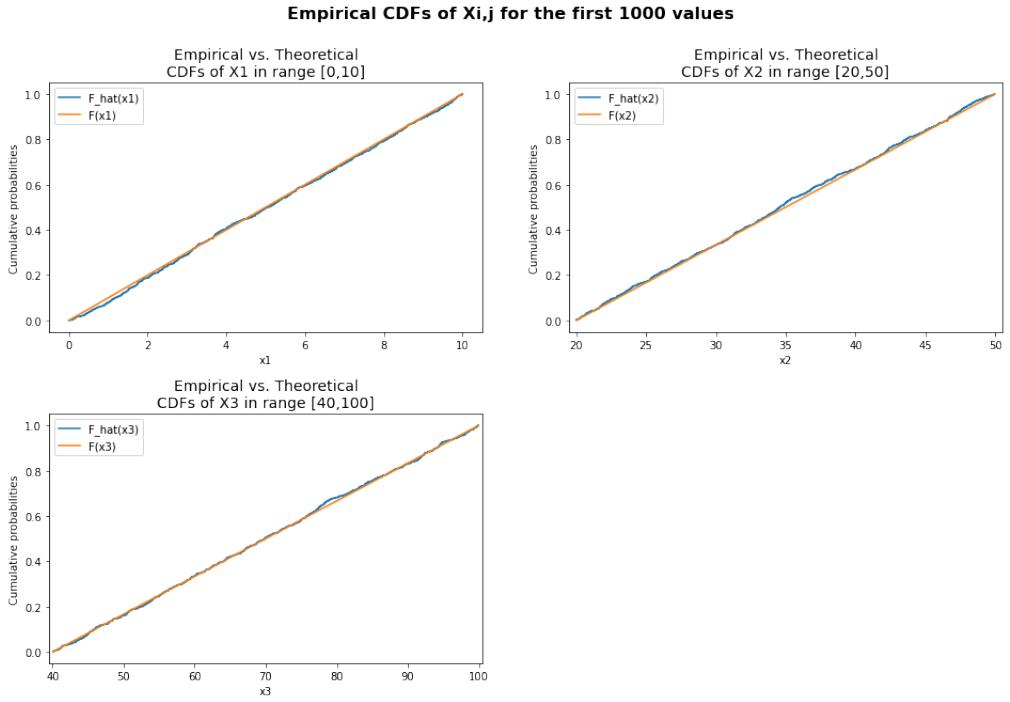

Figure 5: CDF and ECDF for the first 1000 values of the unsorted generated X_i sequences.

Team contribution

Bastien: (25%) coding parts (sample generation and formulas, plotting), provided mathematical justification and preparation of the report.

Ashik: (25%) coded the sorting algorithms, helped in the plotting and did comparison of algorithms.

Kachikwu: (25%) Conclusion regarding the best sorting algorithm (Bucket Sort), calculated the inverse transform method to generate the sequence.

Hongbo: (25%) Helped with the differentiation between theoretical and empirical CDF, and getting to sensible result for the empirical formula and interpretations.