Analysis of Dish Washing

Alessandro Guaresti

December 6th 2022

1 Introduction

Washing dishes is one of my least favorite cleaning activities, but is inevitable when cooking; one of my favorite hobbies. In the context of this class, I thought it would be interesting to see how the run times of hand washing dishes.

2 Modeling

2.1 Dishes

Dishes come in all forms, shapes and sizes. As such, the necessary modeling variable of classification impacts the amount of time a dish takes to handle and wash. From my own kitchen I have forms six possible classifications for dishware: utensils, flat, bowl, cups, pans, and pots. Utensils are your classic eatery tools such as forks and spoons, but it also includes kitchen knives, slotted spoons; any handheld tool used to manipulate food. Flat refers to dishware that is, as the name suggests, flat in nature; think of plates, baking sheets, cutting boards. Bowls are self explanatory, any dishware that has two circles of different radius connecting with a sloped wall. Another aspect of dishes is how messy they will be. Since not all dishes are utilized at the same mess level. Some dishes are more caked on than others. Sometimes you forget to wash dishes for a while and all the food left on them will dry out and stick harder.

2.2 Washing

According to the organization Stop Foodborn Illnesses, it takes around thirty (30) seconds of cleaning and one hundred and seventy (170) degree Fahrenheit water to properly clean a dish. This means that, at a minimum, plates will take 30 seconds of time to clean(TTC). TTC can be defined further as the time it takes to clear, scrub, and rinse. Although these times may vary for different dish classifications and mess level. Now for assumptions: 1. clearing a plate takes on average 5 seconds, 2. at the base scrubbing and rinsing will take 15 seconds each (for a complete 30 seconds)

3 Experiment

A simulated model (done in python) to run a set of 100 randomized dish combinations. All plates are assumed to not be cleared, scrubbed or rinsed. This set of dishes will be cleaned in a couple different ways: 1. dishes cleaned in place as if they were placed in the sink and washed in that order.

Algorithm 1 Wash in place

Require: Dish[100]
  time = 0
  for $i = 1$ to 100 **do**
    time = time + clear(Dish[i])
    time = time + scrub(Dish[i])
    time = time + rinse(Dish[i])
  end for

2. dishes are cleared and sorted by dish classification, then scrubbed and rinsed in their respective groups. There are a few applicable sorting algorithms that I will apply to algorithm 2.

Algorithm 2 Wash with sort

```
Require: Dish[100]
  Sort(Dish)                                  ▷ Scrape will be done while sorting
  for i = 1 to 100 do
      scrub(Dish[i])
      rinse(Dish[i])
  end for
```

3.1 Sort Algorithms in Context

3.1.1 Bucket sort

Bucket sort is the use of categories that dishes have. When sorting dishes into buckets there will be several piles in which the different categories of dishes will be placed into.

Algorithm 3 Dish Bucket Sort

```
Require: Dishes[100]
Require: Create bucket for every dish classification
  for classification in bucket do
      for Dish in classification do
          clear(Dish)
          bucket[Dish.type] = Dish
      end for
  end for
```

3.1.2 Insertion sort

Dishes are capable of being put on top or in between two other dishes until you have a nice sorted pile of dishes.

Algorithm 4 Dish Insertion Sort

```
Require: Dishes[100]
  for i = 1 to 100 do
      key = Dishes[i]
      j = i − 1
      clear(Dishes[i])
      while j ≥ 0 & key < Dishes[j]. do
          Dishes[j+1] = Dishes[j]
          j = j − 1
      end while
      Dishes[j + 1] = Dishes[i]
  end for
```

3.1.3 Merge sort (iterative)

Half of the dishes in the sink are taken and further divided into halves until they can be sorted into order.

Algorithm 5 Dish Merge sort

```
Require: Dishes[100]
  width = 1
  n = len(Dishes)
  while width < n do
      l=0;
      while l < n do
          r = min(l+(width*2-1), n-1)
          m = min(l+width-1,n-1)
          merge(Dishes, l, m, r)
          l += width*2
      end while
      width *= 2
  end while
```

Algorithm 6 Merge

```
Require: Dishes[100], l - left, m - middle, r - right
  n1 = m - l + 1
  n2 = r - m
  L = [0] * n1
  R = [0] * n2
  for  i = 0 to n1 do
      L[i] = Dishes[l + i]
  end for
  for  i = 0 to n2 do
      R[i] = Dishes[m + i + 1]
  end for
  i, j, k = 0, 0, l
  while i < n1 and j < n2 do
      if L[i].ttc ≤ R[j].ttc then
          Dishes[k] = L[i]
          i += 1
      else
          Dishes[k] = R[j]
          j += 1
      end if
      k += 1
  end while
  while i < n1 do
      Dishes[k] = L[i]
      i += 1
      k += 1
  end while
  while j < n2 do
      Dishes[k] = R[j]
      j += 1
      k += 1
  end while
```

3.1.4 Quick sort

Much like merge sort but instead of splitting and forming piles, we take a subset of dishes and make them into a comparative pile where one dish is the basis of comparison.

Algorithm 7 Dish Quick sort

```
Require: Dishes[100]
Require: low - index of lowest part of the array
Require: high - index of the highest part of the array
  if low < high then
      pi = Dish-Partition(Dishes, low, high)
      Dish-Quick-Sort(Dishes, low, pi-1)
      Dish-Quick-Sort(Dishes, pi+1, high)
  end if
```

Algorithm 8 Dish Partition

```
Require: array
Require: low - index of lowest part of the array
Require: high - index of the highest part of the array
  pivot = array[high]
  i = low - 1
  for j = low; j < high; j++ do
      clear(array[j])
      if array[j] ≤ pivot then
          i = i + 1
          Swap(array[i], array[j])
      end if
  end for
  Swap(array[i+1] ,array[high])
  return i+1
```

3.1.5 Selection sort

We take the dishes from the top of the sink pile and compare them from the sorted pile. Then the dish is compared amongst the pile until a proper place is found.

Algorithm 9 Dish Selection sort

```
Require: Dishes[100]
  for i = 0; i < Dishes.length; i++ do
      clear(Dishes[i])
      minI = i
      for j = i + 1; j < Dishes.length; j++ do
          if Dishes[minI] > Dishes[j] then
              minI = j
          end if
      end for
      swap(Dishes[i], Dishes[minI])
  end for
```

4 Results

Having run the simulation one hundred times with six different algorithms under the randomized seed 8675309. Each trial had a sink of different one hundred dish combinations. The following were the result times:

Algorithm	time (seconds)	time (minutes)
Washing in Place	2037.72	33.96
Bucket Sort	2112.72	35.21
Insertion Sort	2654.49	44.24
Merge Sort	2037.72	33.96
Iterative Quick Sort	2164.45	36.07
Selection Sort	3037.69	50.63

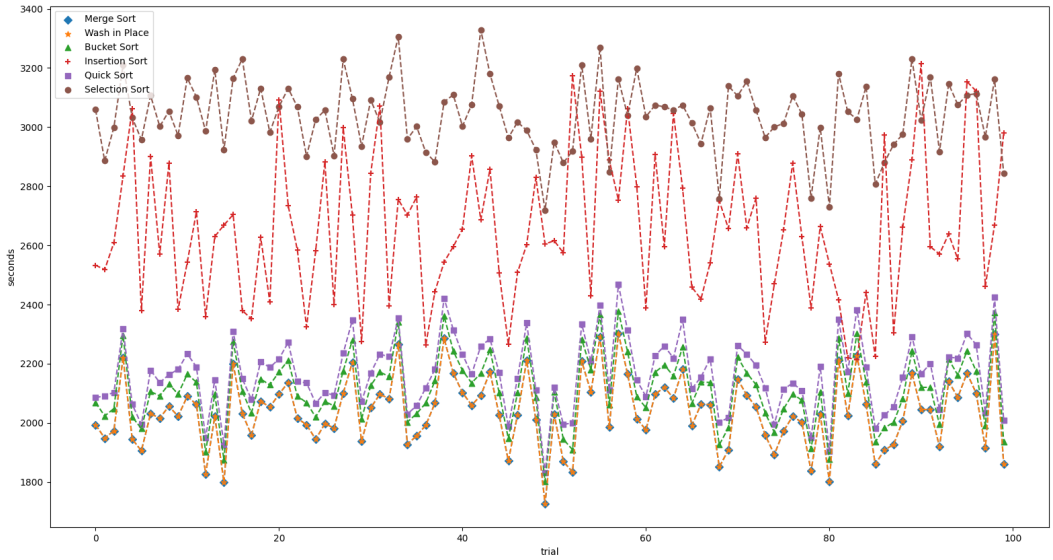

Figure 1: Trial times

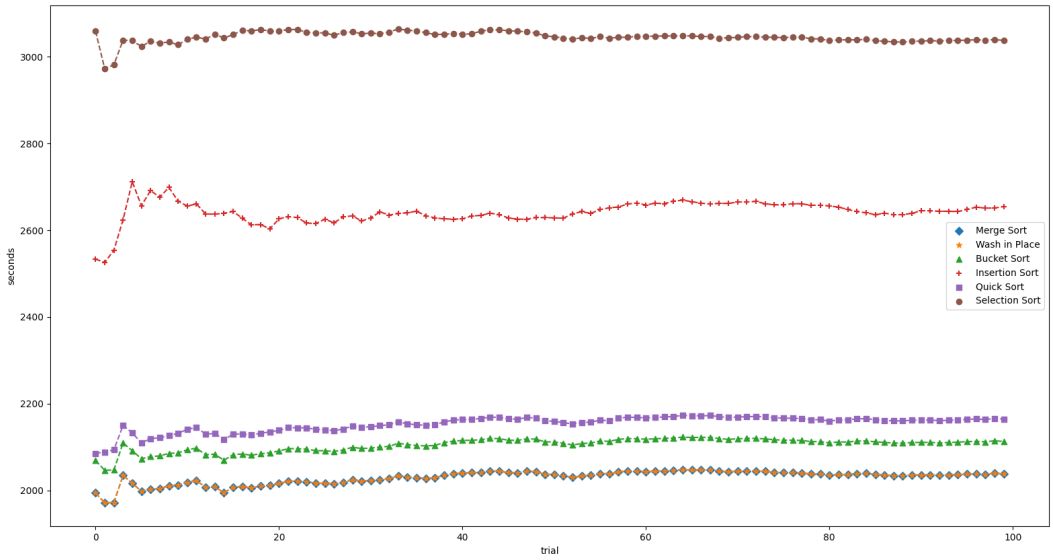

Figure 2: Time Averages

5 Final Conclusions

Washing dishes is an arduous endeavor, and has multiple ways to be done. The simplest and most efficient way to wash dishes is to wash the nearest dish until there are none left. However, the least chaotic and efficient way to wash dishes is to Selection sort them while clearing any caked on debris.

5.1 Further Research

In the future this research could be improved upon by applying more different sorting algorithms and make use of ordering. As well as adjusting the timing of actions within the algorithms to match the average person.