Concepts in Programming

Due: Friday, November 15, 2024, 12:00 PM

Problem Set 4

Problem 1: Collecting the Full Set **[10 pts]**

(a) Experimentally investigate how many times one must randomly select a number from $\{1, 2, \ldots, n\}$ until every number has appeared at least once. For each value of $n = 10$, 100, and 1000, perform 100 independent trials. In each trial, record the total number of selections required to complete the set, and summarize your results by reporting the minimum, maximum, and average number of selections required to observe all numbers. In each experiment, count the number of trials needed.

(b) Extend the experiment in Part (a) to examine how many times the most frequently selected number appears by the end of each trial. For each value of n, compute the minimum, maximum, and average occurrences of the most frequent number across the 100 trials.

Problem 2: Root Finding Methods **[10 pts]**

Let $a \geq 0$ be a real number. A well-known method to compute the square root $\sqrt{a}$ of a is as follows:

Begin with an initial estimate $x_0 = 1$. For $n \geq 1$, compute each subsequent term by the recurrence relation

$$x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}}\right).$$

Repeat this process until x_n is "sufficiently close" to $\sqrt{a}$.

(a) What is the idea behind the formula for x_n?

(b) Implement this method in Python.

(c) Now assume $a > 1$. An alternative approach uses an interval search method:

- Start with a *search interval* $[x, y]$ initialized to $[1, a]$.
- In each iteration, set $z = \frac{x+y}{2}$ as the midpoint of the interval.
- If $z^2 > a$, set $y = z$; otherwise, set $x = z$.
- Repeat until z is "sufficiently close" to $\sqrt{a}$.

What is the underlying idea of this alternative method? Implement it in Python. Which method do you find more effective?

Problem 3: Composite Data Types in Python **[10 pts]**

Consider the following sequence of Python statements. Determine the output of each 'print' statement and identify any statements that produce errors.

```
a = [1, 3, (4, 2)]
b = (a, a, [1, 5, 4, 3])
print(b[1][2])
b[1][2] = "wer"
print(b)
a[2][1]
a[2][1] = "wann"
print(b)
b[2][1]
```

Original TEX template courtesy of W. Mulzer

```
b[2][1] = ["wo", "wie"]
print(b)
b[2] = [2, 4]
print(b)
a = "warum"
print(b)
```