Algorithms: Design and Analysis - CS 412

Weekly Challenge 10: Randomized Algorithms

Ali Muhammad Asad - aa07190

We are going to empirically ascertain the claims about certain expected values of some randomized algorithms.

Running Best A common operation on a sequence of n items is to linearly scan it and maintain a *running best*, i.e. the value that is best so far, as the scan proceeds. The value of the running best when the scan terminates is the best over the entire sequence. We are interested in the number, X, of times that the running best updates during a scan. The value of X for a given value of n depends on the particular permutation of the n items. However, assuming each permutation to be equally likely to appear as the input, the average or expected value of X can be shown to be $O(\ln n)$. (See CLRS 5.1 and 5.2, "The hiring problem")

Collisions in a Hash Table Given an initially empty hash table with n vacant slots, we are interested in the number, X, of items that the hash table can accommodate before a collision occurs. The value of X for a given value of n depends on the items being hashed. However, assuming that an item hashes into any of the slots with equal probability, the expected value of X can be shown to be $O(\sqrt{n})$. (See CLRS 5.4.1, "The birthday paradox")

Collecting Stickers Slanty has released n different types of stickers. Each packet contains a sticker and Ibrahim is keen to collect all the types. He asks his father about the number, X, of packets that he must buy before he has at least one of each type. The value of X for a given value of n depends on the stickers contained in the packets that Ibrahim buys. But his father is a CS professor and assuming each packet to contain any of the n types of stickers with equal probability, he computes the expected value of X as $O(n \ln n)$. (See CLRS 5.4.2, "Balls and bins")

TASKS:

(a) For each experiment below, obtain the average value of X for a given n by conducting several experiments. Plot the resulting X values against n for different values of n, e.g. `range(10**2, 10**5 + 1, 500)`.

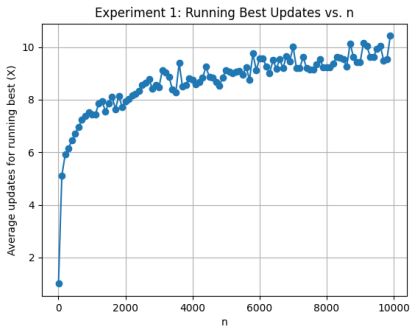
Experiment: For a given n, generate a random permutation of the sequence, $\langle 1, 2, 3, \ldots, n \rangle$, and find X as the number of times that the running best is updated when computing the minimum number in the sequence.

Experiment: For a given n, generate random integers between 1 and n inclusive and find X as the count of unique random numbers generated before the random number is the same as a previously generated random number.

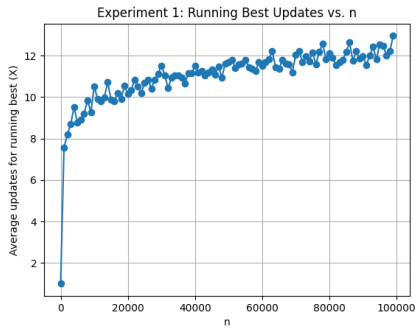
Experiment: For a given n, generate random integers between 1 and n inclusive and find X as the count of random numbers generated until every number from 1 to n has been generated at least once.

(b) Include your plots below along with any notable observations.

Solution: Experiment 1 - Running Best:



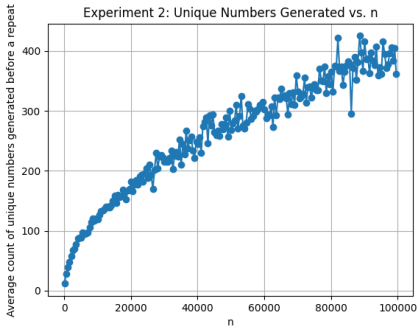
(a) 10000 iterations



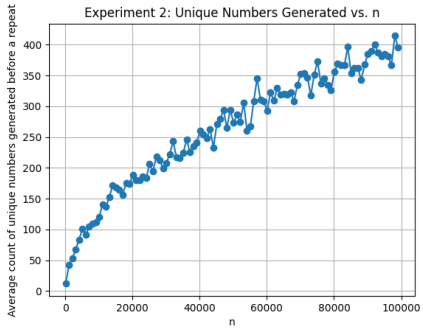
(b) 100000 iterations with step of 1000

From the above figures we can see that the average value of X for increasing iterations follows a logarithmic trend empirically as well which gives weight to the theoretical claim that $X = O(\ln n)$.

Experiment 2 - Collisions in a Hash Table:



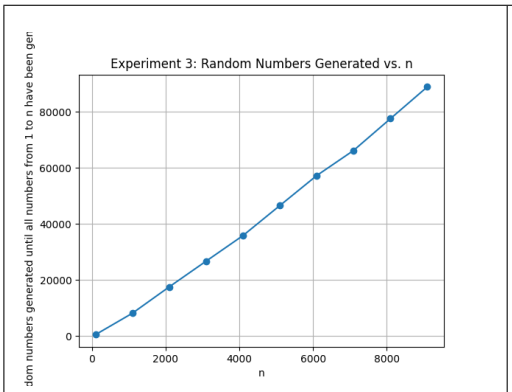
(c) 100000 iterations with step of 500



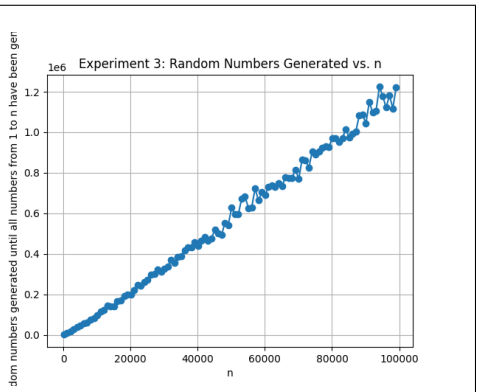
(d) 100000 iterations with step of 1000

From the above figures we can see that the average value of X for increasing iterations follows a square root trend empirically as well which gives weight to the theoretical claim that $X = O(\sqrt{n})$.

Experiment 3 - Collecting Stickers:



(e) 10000 iterations with step of 500



(f) 100000 iterations with step of 1000

From the above figures we can see that the average value of X for increasing iterations follows a $n \ln n$ trend empirically as well which gives weight to the theoretical claim that $X = O(n \ln n)$.

Code Listing that Generates the Above Plots:

```
import random
import matplotlib.pyplot as plt

def running_best_update(n): #Experiment 1
  sequence = list(range(1, n+1))
  random.shuffle(sequence)
  running_best = float('inf')
  updates = 0
  for num in sequence:
    if num < running_best:
      running_best = num
      updates += 1
  return updates

def urn(n): #Experiment 2
  unique_nums = set()
  count = 0
  while True:
    rand_num = random.randint(1, n)
    if rand_num in unique_nums:
      break
    unique_nums.add(rand_num)
    count += 1
  return count

def nuac(n): #Experiment 3
  nums_to_cover = set(range(1, n+1))
  nums_generated = set()
  count = 0
```

```
  while nums_to_cover:
    rand_num = random.randint(1, n)
    if rand_num in nums_to_cover:
      nums_to_cover.remove(rand_num)
    nums_generated.add(rand_num)
    count += 1
  return count

s = 100
e = 100001
step = 500

num_experiments = 100

def e1():
  n_values = []
  avg_updates = []

  for n in range(1, 100001, 1000):
    updates_sum = 0
    print(f"n: {n}")
    for _ in range(num_experiments):
      updates_sum += running_best_update(n)
    avg_updates.append(updates_sum / num_experiments)
    n_values.append(n)

  plt.plot(n_values, avg_updates, marker='o', linestyle='-')
  plt.xlabel('n')
  plt.ylabel('Average updates for running best (X)')
  plt.title('Experiment 1: Running Best Updates vs. n')
  plt.grid(True)
  plt.show()

def e2():
  n_values = []
  avg_counts = []

  for n in range(s, e, step):
    counts_sum = 0
    print(f"n: {n}")
    for _ in range(num_experiments):
      counts_sum += urn(n)
    avg_counts.append(counts_sum / num_experiments)
    n_values.append(n)

  plt.plot(n_values, avg_counts, marker='o', linestyle='-')
  plt.xlabel('n')
  plt.ylabel('Average count of unique numbers generated before a repeat')
  plt.title('Experiment 2: Unique Numbers Generated vs. n')
  plt.grid(True)
  plt.show()

def e3():
  n_values = []
  avg_counts = []

```

```
  for n in range(s, e, step):
    counts_sum = 0
    print(f"n: {n}")
    for _ in range(num_experiments):
      counts_sum += nuac(n)
    avg_counts.append(counts_sum / num_experiments)
    n_values.append(n)

  plt.plot(n_values, avg_counts, marker='o', linestyle='-')
  plt.xlabel('n')
  plt.ylabel('Average count of random numbers generated until all numbers
    from 1 to n have been generated at least once')
  plt.title('Experiment 3: Random Numbers Generated vs. n')
  plt.grid(True)
  plt.show()

e1()
e2()
e3()
```