Rodney Lafuente Mercado

CS181-F22

Assignment #0

Due: 11:59pm EST, Sept. 12, 2022

1. Probability

1.1 For any given bar, let C be the probability that the bar cracks and let B be the probability that the bar has surface bubbles.

We are given that $P(C) = 1/100$, $P(B|C) = 2/300$, $P(B|C^C) = 1/50$, and it can consequently be calculated that $P(C^C) = 99/100$.

Then consequently have, using Bayes Rule,

$$P(C|B) = \frac{P(B|C)P(C)}{P(B)}$$

$$= \frac{P(B|C)P(C)}{P(B|C)P(C) + P(B|C^C)P(C^C)}$$

$$= \frac{(2/100)(1/100)}{(2/300) * (1/100) + (1/50) * (99/100)}$$

$$\boxed{P(C|B) \approx 0.00336 = 0.336\%}$$

Following this calculation, we also get the following: $P(B) \approx 0.01987$, $P(B^C) = 0.98013$.

1.2 In order to answer the question we first need to know the value of $P(C|B^C)$. This can be calculated the following way: we have, by the Law of Total Probability,

$$P(C) = P(C|B^C)P(B^C) + P(C|B)P(B)$$

$$1/100 = P(C|B^C) * 0.98013 + 0.00336 * 0.01987$$

$$\implies P(C|B^C) \approx 0.01013$$

From the fact that $P(C|B) \approx 0.00336$ and $P(C|B^C) \approx 0.01013$, we calculate the expected number of cracked bars:

$$4470 * 0.00336 + (10000 - 4470) * 0.01013 = 71.0381$$

$$\implies \boxed{71 \text{ cracked bars}}$$

1.3 To calculate the expected number of observed bars X before one with peanuts, we use the following observation:

$$E[X] = \sum_{i=1}^{\infty} i \cdot (1/2)^i$$

This expression is stating the fact that the probability of observing i bars before finding one with peanuts is equal to $(N)^{i-1} * (N^C)^1 = (1/2)^{i-1} * (1/2)^1 = (1/2)^i$, where N is the probability of observing peanuts in a any chosen bar. Using mathematical identities, we have

$$\boxed{E[X] = \sum_{i=1}^{\infty} i \cdot (1/2)^i = 2}$$

If we wish to stop only once we have observed two bars with peanuts in a row then we can condition X on three distinct scenarios: the probability that the first bar observed does not contain peanuts, the probability that the first bar and the second bar contain peanuts, and the probability that the first bar contains peanuts and the second does not (these three cases are mutually exclusive and collectively exhaustive). These events have probabilities of $1/2, 1/4$, and $1/4$ respectively. This is expressed as

$$E[X] = (1/2) * E(X|N^C) + (1/4) * E(X|NN) + (1/4) * E(X|NN^C)$$

we can simplify this by observing that $E(X|N^C) = 1 + E(X)$, $E(X|NN^C) = 2 + E(X)$, and $E(X|NN) = 2$:

$$E[X] = (1/2) * (1 + E(X)) + (1/4) * 2 + (1/4) * (2 + E(X))$$

Solving this equation yields

$$\boxed{E[X] = 6}$$

2. Algorithms

We analyze the running time of each step in the algorithm.

Step (1) runs in $O(k)$ time (we create an empty array, which is a constant time operation, k times), which is equal to $O(n)$ time as $k = \Theta(n)$. Step (2) takes $O(n)$ time as we are performing a constant time operation for each element of the input array of length n. Step (3) is where things get tricky. We know that each insertion sort on each array takes, in the worst case, $O(n_i^2)$ steps, where n_i is the number of elements in the ith array. In order to calculate the average running time of each insertion sort we can calculate running time for the expected length of the array, squared (denoted $E[n_i^2]$). We do this in the following manner:

Let X_{ij} be the indicator variable equal to 1 iff the jth element of the input array is inserted into the ith array in step (2) of the algorithm. Then we have

$$\begin{aligned} E[n_i^2] &= E\left[\sum_{i=1}^{n} X_{ij}^2\right] + E\left[\sum_{i=1}^{n}\sum_{k\neq i} X_{ij}X_{ik}\right] \\ &= \sum_{i=1}^{n} E[X_{ij}^2] + \sum_{i=1}^{n}\sum_{k\neq i} E[X_{ij}X_{ik}] \end{aligned}$$

Given that each X is an indicator variable, we have $E[X_{ij}^2] = E[X_{ij}] = 1/k$. The $1/k$ is derived from the fact that the probability that any given value in the input array is placed into any one of the k possible arrays is $1/k$. Likewise, we have $E[X_{ij}X_{ik}] = 1/k^2$ as the product of these two indicator variables is 1 iff two different elements from the input array get placed into the same array, which has a probability $1/k^2$ of occurring. Substituting these values in the aforementioned equation gives

$$\begin{aligned} E[n_i^2] &= \sum_{i=1}^{n} 1/k + \sum_{i=1}^{n}\sum_{k\neq i} 1/k^2 \\ &= n/k + (n(n-1))/k \\ &= n/k + n^2/k^2 + n/k^2 \end{aligned}$$

Therefore, we have that the average runtime of EACH each insertion is $O(n/k + n^2/k^2 + n/k^2)$. Since $k = \Theta(n)$, the biggest term is equivalent to 1 and this is equal to constant time ($O(1)$). The total average work for step (3) is, therefore, k constant time operations, which is an average running time of $O(k) = O(n)$.

Finally, the worst case running time of step (4) is $O(k) = O(n)$ as we are performing a constant time concatenation for each of the k arrays.

Since steps (1), (2), and (4) have a worst case running time of $O(n)$, and step (3) has an average running time of $O(n)$, the algorithm as a whole has an average running time of $O(n)$.

3. Programming Warm-Up

   Implemented in pset0.py

4. Encryption Algorithms

   4.1 Caesar Cipher

   (a) Implemented in pset0.py

   (b) The issue with Bob's approach is that the double-encryption is actually equivalent to encrypting only once using the cipherkey with corresponding translation equal to the modular sum of both translations for t and h. It doesn't matter what order Bob makes those encryptions

   4.2 Caesar Cipher Auto-Decryption

   Implemented in pset0.py

   4.3 Vigenere Cipher

   (a) Implemented in pset0.py

   (b) This is a stronger method of encryption as information on the true value of one character gives zero information on the true value of other characters (whereas in the other two ciphers, the information given is complete). The distribution of letters using this cipher would be the same for each letter.

   4.4 Collaboration, Calibration, and References

   (a) I did not work with anyone nor use any references.

   (b) I spent roughly six hours on this problem set.