Harvey's Notes I - Chapter 01

Mostafa Touny

November 15, 2023

Contents

Exercises

Ex. 1

Part I

```
def randElement(A[1..n])
  X = random( [n] )
  return A[X]
```

Since the probability space is uniform, For event $M = \{\frac{n}{4} + 1, \ldots, \frac{3}{4}n\}$, $Pr[M] = \frac{1}{n} \cdot |M| = \frac{1}{n} \cdot \frac{n}{2} = \frac{1}{2}$.

Part II

Additionally we certify if the randomly generated element is in middle half.

```
def randElement2(A[1..n])
  # Select a random element of A
  k = randElement(A)

  # Certify whether it is in middle half

  # count values less and greater
  countLess = countGreater = 0
  for i in 1..n:
    if A[i] < k
      countLess = countLess + 1
    else if A[i] > k
      countGreater = countGreater + 1

  # check if k is between first and forth quarters
  if (countLess >= n/4) and (countGreater >= n/4)
    return k
  return FAIL
```

Let R be the algorithm's output. $R = FAIL$ if and only if $\neg M$. So $Pr[R = FAIL] = Pr[\neg M] = 1 - Pr[M] = 1 - \frac{1}{2} = \frac{1}{2}$.

Part III

We repeat until the probability is upperbounded by 0.01.

```
def randElement3(A[1..n])
  # repeat 7 times
  for i in 1..7
```

```
    # generate a random element
    out = randElement2(A[1..n])

    # if the number is certified to be correct return it
    if out != FAIL
      return out

  # if 7 trials failed
  return FAIL
```

Setting $0.5^x = 0.01$ we get $x = \log_{1/2} 0.01 = \dfrac{\log_2 100^{-1}}{\log_2 2^{-1}} = \dfrac{(-1)\log_2 100}{(-1)\log_2 2} = \log_2 100 \leq \log_2 128 = \log_2 2^7 = 7$

Let R be the algorithm's output, and let R_i be the output of subroutine *randElement2* in iteration i. Then $R = FAIL$ if and only if $R_1 = FAIL \wedge \cdots \wedge R_7 = FAIL$. We know $Pr[R_i = FAIL] = \frac{1}{2}$ and R_i are pairwise independent. We conclude $Pr[R = FAIL] = Pr[R_1 = FAIL \wedge R_2 = FAIL \wedge \cdots \wedge R_7 = FAIL] = \left(\frac{1}{2}\right)^7 \leq 0.01$.

Ex. 2

Part I

Trivial.

Part II

Hint. By Dr. I. El-Shaarawy, Not to skip *Part I*, and to observe the pattern in the following example. It signals the answer is 2^k if $x = 0$ and 2^{k-1} otherwise.

Binary Number	**Count of Even Parity**
000	8
001	4
010	4
011	4
100	4
101	4
110	4
111	4

Lemma 1. The zero $0 = \underbrace{00\ldots0}_{k \text{ times}}$ counts 2^k numbers of even parity.

Trivially, $BitwiseAnd(0, x) = 0$ for any binary number $x \in [2^k]$, and $Parity(0) = 0$.

Now we can focus on $x \neq 0$.

Definition 2. Given x denote indices of 1-bits by *1-bits-indices*.

Lemma 3. *1-bits-indices* decide the parity.

Observe for any $r \in [2^k]$.

$$BitwiseAnd(x_i, r_i) = \begin{cases} 0 & \text{if } x_i = 0 \\ r_i & \text{if } x_i = 1 \end{cases}$$

So we can restrict our focus only on *1-bits-indices* to compute the parity. In other words

$$Parity(BitwiseAnd(x, r)) = \begin{cases} 0 & \text{if } r \text{ has even 1 bits in } \textit{1-bits-indices} \\ 1 & \text{if } r \text{ has odd 1 bits in } \textit{1-bits-indices} \end{cases}$$

Lemma 4. The number of *k-length* strings containing even number of 1 bits in *1-bit-indices* is 2^{k-1}.

Define a bijection

$$f : \{\text{strings of even 1-bits in } \textit{1-bits-indices}\} \to \{\text{strings of odd 1-bits in } \textit{1-bits-indices}\}$$

Mapping a binary string to the same string but with last bit in *1-bit-indices* flipped. If that bit is s_m, Then $f(s_1 s_2 \dots s_k) = s_1 s_2 \dots \overline{s_m} \dots s_{k-1} s_k$. It follows domain and range have the same cardinality, and since they partition the set of *k-length* strings, the result follows.

Theorem 5. Fixing any binary $x \neq 0$, Among all $r \in [2^k]$, Exactly half of them yield even parity, i.e $Parity(BitwiseAnd(x, r)) = 0$.

Corollay 6. Given $x \in [2^k]$, The number of zeros in the vector mentioned in question is

$$\begin{cases} 2^k & \text{if } x = 0 \\ 2^{k-1} & \text{if } x \neq 0 \end{cases}$$