13 Algorithms & Data structures Assignment 13

I Input: an integer n
Output: random number between 0 and $2^n - 1$

Algorithm:

Random(n)

① Run Coin n times so we have n 0's or 1's at the end of all runs.

② With these n bits, we can create a binary number and convert it to decimal representation.

This creates a binary number of n bits, with n bits we can represent the decimal numbers 0 to $2^n - 1$.

The probability of Coin being 1 is:

$$Pr(1) = \frac{1}{2} = Pr(0) \quad (\text{of Coin})$$

The probability of two coin tosses is:

$$Pr(\text{Coin toss \#1}) \cdot Pr(\text{Coin toss \#2}) = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$$

Because we do n coin tosses, all probabilities are ~~w~~being multiplied:

bit 1 bit 2 bit 3 ...

$\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \ldots$

This creates a formula:

$Pr(\text{a number between } 0 \text{ and } 2^n - 1) =$

$$\left(\frac{1}{2}\right)^n = \frac{1}{2^n}$$

II

i. For the algorithm to always return the correct answer, we must use Det-prime somewhere.

Input: integer n
Output: boolean that is true if n is a prime, false otherwise.

We need to execute Rand-prime x times before Det-prime to filter out all non-prime numbers.

If a numbe is non-prime and we execute the algorithm Rand-Prime n times, the probability that we get the wrong answer is:

$$\left(\frac{1}{2}\right)^n$$

The probability we get the good answer is:

$$1-\left(\frac{1}{2}\right)^n$$

This creates a formula for the average running time if a number is non prime:

$$t_{avg} = \underbrace{\left(\frac{1}{2}\right)^n \cdot (100+x)}_{\text{wrong answer}} + \underbrace{\left(1-\left(\frac{1}{2}\right)^n\right) \cdot x}_{\text{good answer}}$$

we take x as "~~best~~ worst"

The least average running time is when executing Rand-Prime 6 times

Algorithm:

1. Execute Rand-Prime 6 times, if one of them returns "not-prime" return "not-prime".
2. Execute Det-prime and return its result.

2. The running time of the algorithm if the number is prime is $100\,T(n) + 6T(n) = 106\,T(n)$

3. The running time of the algorithm if the number is non prime is:

$$\left(\tfrac{1}{2}\right)^6 \cdot (106\,T(n)) + \left(1 - \left(\tfrac{1}{2}\right)^6\right) \cdot T(n)$$

$$\downarrow \qquad\qquad \downarrow$$

$$\approx 1{,}65\,T(n) + 5{,}9\,T(n) \approx 7{,}55\,T(n)$$

The average running time is $8\,T(n)$

III If we pick each node color at random, the probability that an edge e satisfies the condition is $2/3$.

Input: graph $G = (V, E)$

Output: colored graph $G = (V, E)$ with at most three colors

Algorithm:

for each $v \in V$:

pick a random color and assign it to v

The probability that the condition is satisfied is:

$$E[\text{satisfied edges}] = \sum_{e \in E} E[e \text{ is a satisfied edge}]$$

$$= P(e \text{ is satisfied}) \cdot |E|$$

$$= \frac{2}{3} \cdot |E|$$

because $|E|$ is always large than or equal to c^*, the property is satisfied.

IV $X_k = 1$ if bid k updates b^*

$X_k = 0$ if bid k does not update b^*

$$E[\#\text{ of } b^* \text{ is updated}] = \sum_{1 \le k \le n}^{k} (E(X_k))$$

$$= \sum_{1 \le k \le n}^{k} (P_r(b_k \text{ updates } b^*))$$

$$= \sum_{1 \le k \le n}^{k} (1/k)$$

$P_r(b_k \text{ updates } b^*) = 1/k$ because: (next page)

~~with $n = 2$, there are two possibilities:~~

~~$b_1 < b_2$ or $b_2 < b_1$~~

with $n=1$, b^* gets updated once:

$b_1 > b^* = 0$ because $b_1 > 0$

with $n=2$, b^* gets updated $\approx 1\frac{1}{2}$:

$b_1 > b^* = 0$ and $b_1 < b_2$ or $b_1 > b_2$

...

$$E[\text{\#}\, b^* \text{ updated}] = \sum_{1 \le k \le n}^{n} (1/k) = 1 + 1/2 + 1/3 + \ldots + 1/n$$

V Let's look at some steps:

$k=1$

depth leafs $= 1$

$P_r = 1$

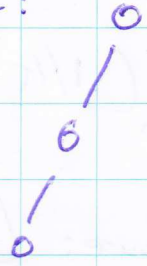
$k=2$

leafs $= 1$

$P_0 = 1$

Avg $= 1$

$k=3$ either:

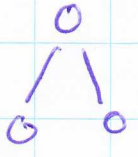


leafs $= 1$

$P_r = 1/2$

Avg $= 1/2 \cdot 1 + 1/2 \cdot 2 = 1\frac{1}{2}$

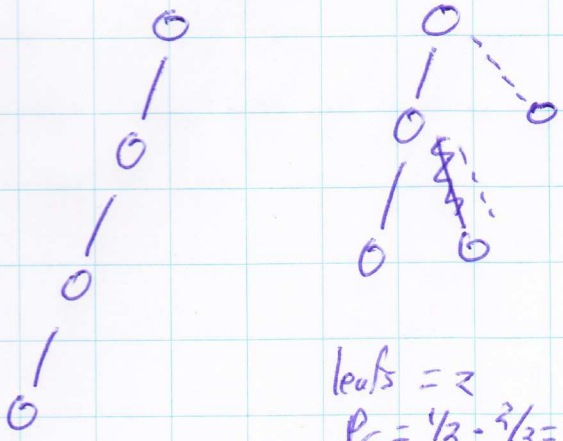
or:



leafs $= 2$

$P_r = 1/2$

$k = 4$



leafs = 1
$P_r = 1/2 \cdot 1/3 = 1/6$

leafs = 2
$P_r = 1/2 \cdot 2/3 = 2/6$

leafs = 2
$P_r = 1/2 \cdot 2/3 = 2/6$

leafs = 3
$P_r = 1/2 \cdot 1/3 = 1/6$

$$Avg = 1/6 \cdot 1 + 4/6 \cdot 2 + 1/6 \cdot 3 = 1/6 + 8/6 + 3/6 = 12/6 = 2$$

This continues ...

At each step, there is a 50% chance that the tree will grow in depth or in width.

- ↓ no new leaf
- ↳ new leaf

We can create a formula:

$$\#leafs = \begin{cases} 1 & \text{if } k \leq 2 \\ 1 + \frac{k}{2} & \text{otherwise} \end{cases}$$