- Let's now take a detour to some practical problems of probability.

Uniform sampling from $[0 .. m-1]$

- Given an integer m (& unbiased coin), design an algorithm to pick random $x \in [0...m-1]$.

Idea: Toss the coin $k := \lceil \lg m \rceil$ many times?
The binary string might give $x > m-1$?

Algo: 0) $k := \lceil \log_2 m \rceil$.

1) Toss the coin k-times to get $\underline{X} := \sum_{i=0}^{k-1} X_i \cdot 2^i$

($\underline{X_i} = 0$ or 1 in i-th toss.)

2) If $X \geq m$, then GOTO (1).

else output X.

$\triangleright$ $\forall t \in [0..m-1]$: $P(\text{Output is } t) = P(X=t \mid X \in [0..m-1])$

$$= \frac{P(X=t \wedge X \in [0..m-1])}{P(X \in [0..m-1])} = \frac{1/2^k}{m/2^k} = \frac{1}{m}.$$

uniform prob. $\forall t$!

$\triangleright$ $\forall t \notin [0..m-1]$:

$P(\text{Output } t) = 0$.

- But, the 'GOTO' may execute only many times!

- $E[\#\text{times (1) is executed}] =: \mu$. Let $\delta := 1 - \frac{m}{2^k} \leq \frac{1}{2}$.

$$\Rightarrow \mu = \frac{m}{2^k} \cdot \underline{1} + \left(1 - \frac{m}{2^k}\right) \cdot \frac{m}{2^k} \cdot \underline{2} + \left(1 - \frac{m}{2^k}\right)^2 \cdot \frac{m}{2^k} \cdot \underline{3} + \cdots$$

$$\Rightarrow \frac{2^k}{m} \cdot \mu = 1 + \delta \cdot 2 + \delta^2 \cdot 3 + \cdots$$

$$\Rightarrow 2^k \mu \delta / m = \delta \cdot 1 + \delta^2 \cdot 2 + \cdots$$

$$\Rightarrow \frac{2^k}{m} \mu \cdot (1 - \delta) = 1 + \delta + \delta^2 + \cdots = \frac{1}{1 - \delta} \quad \text{(Geometric sum)}$$

$$\Rightarrow \mu = 2^k / m < 2.$$

- The algo. is <u>expected</u> to stop in $2k = \underline{O(\lg m)}$ <u>tosses</u>!

Sampling k numbers from $[0 \ldots m-1]$

- Given k, m; you want to pick a random subset $S \subseteq [0 \ldots m-1]$ of size $|S| = k$.

Idea: Keep picking an element, till you get a new one.

Algo: 0) $S \leftarrow \phi$;

1) for $i \in [k]$ {

2) $X \leftarrow$ uniformly picked in $[0 .. m-1]$;

3) If $X \in S$, then GOTO (2);

else $S \leftarrow S \cup \{X\}$;

}

4) Output S;

- Let's analyze the <u>iterations</u> $i=1,2$:

▷ Let $t_1 \neq t_2 \in [0..m-1]$. $P(S=\{t_1,t_2\})$

$$= P(t_1 \in S)\cdot P(X=t_2 \mid t_1 \in S) + P(t_2 \in S)\cdot P(X=t_1 \mid t_2 \in S)$$

$$= \frac{1}{m}\cdot\frac{1}{m-1} + \frac{1}{m}\cdot\frac{1}{m-1}$$

$$= 2/m(m-1) = 1/\binom{m}{2}.$$ [<u>Uniform</u> distr.!]

Similarly:

▷ $P(S=\{t_1,\dots,t_k\}) = 1/\binom{m}{k}$.

▷ $E[\#\text{iterations for an } i] = m/(m-i+1)$.

$\Rightarrow E[\#\text{steps in the algo.}] = \sum_{i\in[k]} m/(m-i+1)$

$$= m\cdot\left(\frac{1}{m}+\frac{1}{m-1}+\dots+\frac{1}{m-k+1}\right) \approx m\cdot\log m \quad (\text{for } \underline{\text{large}}\ k)$$

$$\approx k \quad (\text{for } \underline{\text{small}}\ k)$$

Biased Coin-toss

- Given integers α, m s.t. $0 \le \alpha \le m-1$. Simulate a coin-toss with $P(H) = \underline{p} := \alpha/m$.
- Simulate this using an $\underline{\text{unbiased}}$ coin?
  ↳ looks difficult!

Idea: Sample $X \in [0 \dots m-1]$ & $\underline{\text{output } H}$ if $X \in [0 \dots \alpha-1]$.

Algo: 1) $X \leftarrow$ rnd. number in $[0 \ldots m-1]$ ← use unbiased coin.
2) If $X < \alpha$, then Output $\underline{H}$.
else " $\underline{T}$.

▷ # rnd. bits used $= O(\lg m)$.

▷ $P(\text{output}=H) = P(X<\alpha) = \dfrac{\#[0 \ldots \alpha-1]}{\#[0 \ldots m-1]} = \dfrac{\alpha}{m}$.

<u>Uniform Sampling a permutation</u> of $[n]$

- Given n, you want to pick a permutation, say as a string $\underline{S}$. Eg. 132 of $[3]$.

<u>Idea</u> - keep picking an element $X \in [n]$ that has <u>not</u> been picked before; <u>grow</u> string S by X.

<u>Algo:</u>

0) $S \leftarrow$ empty-string;

1) For $i \in [n]$ {

   2) $X \leftarrow$ rnd number in $[n]$;

   3) If X is in S, then GOTO (2);

      else $S \leftarrow S \cdot X$ (<u>Append</u> X in the end)

   }

4) Output S;

- Analyse the first two 'for'-iterations:

▷ Let $t_1 \neq t_2 \in [n]$. $P(S = t_1 t_2) = P(S = t_1) \cdot P(X = t_2 \mid S = t_1)$

$$= \frac{1}{n} \cdot \frac{1}{n-1}.$$

uniform distr.

⇒ By induction on i:

▷ $P(S = t_1 \cdots t_n) = 1/n(n-1)(n-2)\cdots 1 = 1/n!$.

▷ $E[\#\text{steps}] = \sum_{i=1}^{n} \frac{n}{n-i+1} = n \cdot \left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1}\right) \approx n \cdot \log n$

almost linear time!