Search-to-Decision Reduction for Random Local Functions

Low-Complexity Cryptography Workshop

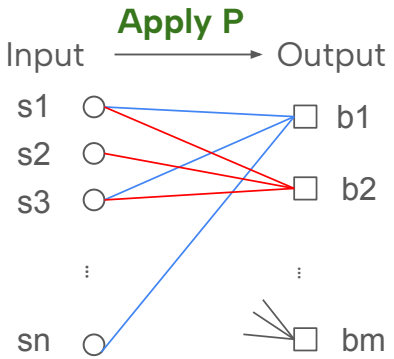
Kel Zin Tan

National University of Singapore

Random Local Function

Define function $f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ where m > n, by two components

- P is a fixed d-ary predicate
  $P : \mathbb{F}_2^d \to \mathbb{F}_2$
- G is a random d-right-regular bipartite ordered graph
  $G = (S_1, S_2, \ldots, S_m) \quad S_i = (j_1, \ldots, j_d)$



The i-th output of the function: Apply P to the neighbours of i-th output node

$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Motivation 1

Fix a predicate $P : \mathbb{F}_2^d \to \mathbb{F}_2$ (Balanced)

Randomly sample secret input s and Graph G.

$$f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$$
$$P : \mathbb{F}_2^d \to \mathbb{F}_2$$
$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Cryptographic Application

- **Pseudorandom Generator (PRG):**
  Distinguish output from random
- **One-way Function (OWF):**
  Given output and G, recover s

Random Local Function introduced by Goldreich as OWF [Gol00]

Key Appeal

Implementable in NC0 (Constant depth circuit)

Motivation 2

Natural **Average Case Problem**

Rich literature on attacks/hardness evidence
[AHI05, CEMT09, BQ09, Its10, OW14, AL16, App16]

$$f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$$
$$P : \mathbb{F}_2^d \to \mathbb{F}_2$$
$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Analog of Random **Constraint Satisfaction Problems (CSP)**

“Deterministic Noise” version for **Sparse LPN (k-LIN)**

$$P(s_1, s_2, s_3, s_4, s_5) = \underbrace{s_1 + s_2 + s_3}_{\text{Linear Part}} + \underbrace{s_4 s_5}_{\text{Noise}}$$

Sparse LPN (Digress)

Set sparsity $k = O(1)$

Sample Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$, s.t. each row has **exactly k** non-zero entries

Sample Secret $s \leftarrow \mathbb{F}_2^n$ Sample Error $e \leftarrow \mathrm{Bern}(0.1)^m$

Recover s from $(A, As + e)$

Key Appeal

- "Harder" than Random Local Function
- With s balanced, random local Function reduces to Sparse LPN

Problem: Non-trivially sample Bernoulli in NC0 from uniform seed is tricky

Search-to-Decision Reduction

$$f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$$
$$P : \mathbb{F}_2^d \to \mathbb{F}_2$$
$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Predicate P is public (Assume balance)

Search: Given $(G, f_{G,P}(s))$, find s

Decision: Given G, distinguish $f_{G,P}(s)$ from random

Search-to-Decision Reductions implies

Search Hardness => Decision Hardness

One-wayness => Pseudorandomness

Warmup 1

Derive Search-to-Decision reduction for Random Local Function with ***sensitive*** predicate

A predicate $P : \mathbb{F}_2^d \to \mathbb{F}_2$ is ***sensitive*** if there exists a full influence variable

> $P : \mathbb{F}_2^d \to \mathbb{F}_2$ is ***sensitive*** if there exists $Q : \mathbb{F}_2^{d-1} \to \mathbb{F}_2$ s.t:
> $$P(s_1, \ldots, s_d) = s_i + Q(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_d)$$

Sensitive P: $P(s_1, s_2, s_3, s_4, s_5) = s_1 + s_2 + s_3 + s_4 s_5$

Non-sensitive P: $P(s_1, s_2, s_3, s_4, s_5) = s_1 s_2 + Maj(s_3, s_4, s_5)$

Warmup 2

$$f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$$
$$P : \mathbb{F}_2^d \to \mathbb{F}_2$$
$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Derive Search-to-Decision reduction for Random Local Function with ***sensitive*** predicate

WLOG assume first input is sensitive, d = 3

$$P(x_1, x_2, x_3) = x_1 + Q(x_2, x_3)$$

Suppose $f_{G,P}(s)_i = P(s_3, s_2, s_4)$

What if first input is change? $P(s_1, s_2, s_4)$

- If $s_1 = s_3$, then $P(s_1, s_2, s_4) = P(s_3, s_2, s_4)$
- If $s_1 \neq s_3$, then $P(s_1, s_2, s_4) \neq P(s_3, s_2, s_4)$

Warmup 3

Only true when first variable is sensitive

- If $s_1 = s_3$, then $P(s_1, s_2, s_4) = P(s_3, s_2, s_4)$
- If $s_1 \neq s_3$, then $P(s_1, s_2, s_4) \neq P(s_3, s_2, s_4)$

Example when variable not sensitive: $P(x_1, x_2, x_3) = x_1 + x_2 \cdot x_3$

Flip second variable does not guarantee flip in output $P(1,0,0) = P(1,1,0) = 1$

Key Idea [App12]:

Use the first variable to predict relationship between secret bits

Key Idea [App12]

Use the first variable to predict relationship between secret bits

A decision algorithm D for a PRG of length m with adv $\mathcal{E}$

↓ Yao's Theorem (Hybrid argument)

A predictor that given (i-1) outputs of PRG, predict i-th output with prob ε/m

Randomly re-select the first variable

$$S_i = (3, 2, 4) \xrightarrow[j \leftarrow [n]]{} S_i' = (j, 2, 4) \longrightarrow$$

Use Predictor to predict

$$y = P(j, 2, 4)$$

If y equals the original i-th output of PRG. Then s3 = sj. Otherwise s3 != sj

Final Part [App12]

Randomly re-select the first variable

$$S_i = (3, 2, 4) \xrightarrow[j \leftarrow [n]]{} S'_i = (j, 2, 4) \longrightarrow$$

Use Predictor to predict

$$y = P(j, 2, 4)$$

If y equals the original i-th output of PRG. Then s3 = sj. Otherwise s3 != sj

Predictor may fail so this gives noisy value of $s_3 \oplus s_j$

Amplify by repetition and solve noisy 2-XOR

Require $O(m^3/\varepsilon^2)$ samples to solve Search using Decision

Result Summary

Decision algorithm with advantage $\mathcal{E}$ and m samples

↓ Construct

Search Algorithm with with advantage $\Omega(\varepsilon)$ and M samples (M > m)

	[App12]
Sensitive	$M = O(m^3/\varepsilon^2), m = poly(n)$
General	$M = O(n), m = O(n)$

Improvement from [BRT25] on sensitive predicate

$$M = O(mn/\varepsilon^2), m = poly(n)$$

Reduction [BRT25]

A more direct approach, does not use Next-Bit Unpredictability Theorem

Different predicate for Decision and Search

Search (d variables): $P(x_1, \ldots, x_d) = x_1 + Q(x_2, \ldots, x_d)$

Decision (d-1 variables): $Q(x_2, \ldots, x_d)$

Similarly try to predict relationship between secret bits

Key Idea [BRT25]

$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Group samples by first variable

$$G_1 = \begin{matrix}(1, 4, 2, \ldots)\\ (1, 3, 1, \ldots)\\ (1, 5, 4, \ldots)\end{matrix} \qquad G_3 = \begin{matrix}(3, 2, 1, \ldots)\\ (3, 1, 4, \ldots)\\ (3, 5, 3, \ldots)\end{matrix} \qquad G_i = \begin{matrix}(i, 1, 2, \ldots)\\ (i, 3, 4, \ldots)\\ (i, 1, 5, \ldots)\end{matrix}$$

Collect $O(nm)$ samples, then each Group G_i on expectation has O(m) samples

To know if s1 = s3 or s1 != s3:

1. Mix samples from G1 and G3 (about half-half)
2. Remove first variable and query decision oracle

Key Idea [BRT25]

$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

To know if s1 = s3 or s1 != s3:

1. Mix samples from G1 and G3 (about half-half)
2. Remove first variable and query decision oracle

$$G_1 = \begin{matrix}(1,4,2,\ldots)\\(1,3,1,\ldots)\\(1,5,4,\ldots)\end{matrix} \longrightarrow \begin{matrix}(4,2,\ldots)\\(5,3,\ldots)\\(2,1,\ldots)\\(5,4,\ldots)\end{matrix} G' \longleftarrow G_3 = \begin{matrix}(3,2,1,\ldots)\\(3,1,4,\ldots)\\(3,5,3,\ldots)\end{matrix}$$

- If $s_1 = s_3 = 0$, get exactly $(G', f_{G',Q}(s))$
- If $s_1 = s_3 = 1$, get exactly $(G', f_{G',Q}(s) + 1^m)$
- If $s_1 \neq s_3$, get exactly (G', random)

Final Part [BRT25]

- If $s_1 = s_3 = 0$, get exactly $(G', f_{G',Q}(s))$
- If $s_1 = s_3 = 1$, get exactly $(G', f_{G',Q}(s) + 1^m)$
- If $s_1 \neq s_3$, get exactly (G', random)

Decision: Given G, distinguish $f_{G,Q}(s)$ from random

Search Algorithm

1. Collect $O(nm)$ samples and group them
2. Guess s1
3. Mix G1 and Gi for all i > 1
4. Oracle can predict whether s1 = si with advantage ε

Result Summary

Decision algorithm with advantage ε and m samples

↓ Construct

Search Algorithm with with advantage $\Omega(\varepsilon)$ and M samples

	[App12]	[BRT25]
Sensitive	$M = O(m^3/\varepsilon^2), m = poly(n)$	$M = O(mn/\varepsilon^2), m = poly(n)$
General	$M = O(n), m = O(n)$	x

What about general predicate with polynomial samples?

[TV26] Search-To-Decision reduction with ***any*** predicate and **m = poly(n)**

Technique Overview

Search Problem : Given $(G, f_{G,P}(s))$, find s

Goal: Construct Search Algorithm Given Decision Oracle

Search Algorithm

What is s_i ?

↓

Apply **Transformation T** on Instance

If $s_1 = s_i$ ↙ ↘ If $s_1 \neq s_i$

Instance close to Planted

Instance close to Random

Use Decision Oracle to distinguish these two cases

Transformation

$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d) \quad j \in [1, n]$$

Objective

- If $s_1 = s_i$, then $(T(G), f_{G,P}(s))$ looks like $(G, f_{G,P}(s))$
- If $s_1 \neq s_i$, then $(T(G), f_{G,P}(s))$ looks like (G, random)

Transformation $T_{a,b}$ (a and b in [n])

$$G = \begin{matrix} S_1 = (2,3,4) \\ S_2 = (1,3,5) \\ S_3 = (1,2,2) \\ S_4 = (5,3,4) \end{matrix}$$

Transformation T on a = 1 and b = 2

T on a = 1 and b = 2 $\longrightarrow$ $T_{a,b}(G) =$

$$\begin{matrix} S'_1 = (1,3,4) \\ S'_2 = (2,3,5) \\ S'_3 = (1,2,1) \\ S'_4 = (5,3,4) \end{matrix}$$

- All values unchanged except for a and b
- **Value a and Value b swap with probability half**

Key Property 1

- If $s_1 = s_i$, $T_{1,i}$ has **no effect** on the distribution

$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Transformation

Transformation T on a = 1 and b = 2

$$G = \begin{matrix} S_1 = (2,3,4) \\ S_2 = (1,3,5) \\ S_3 = (1,2,2) \\ S_4 = (5,3,4) \end{matrix} \xrightarrow{\text{T on a = 1 and b = 2}} T_{a,b}(G) = \begin{matrix} S'_1 = (1,3,4) \\ S'_2 = (2,3,5) \\ S'_3 = (1,2,1) \\ S'_4 = (5,3,4) \end{matrix}$$

- All values unchanged except for a and b
- **Value a and Value b swap with probability half**

Example: $P(s_1, s_i, s_5, s_3) = P(s_1, s_1, s_5, s_3)$

- **Output remains the same**

$$f_{G,P}(s) = f_{T_{1,i}(G),P}(s)$$

- **Graph stays Random**

$$T_{1,i}(G) \approx G$$

Distribution is the same

$$\Rightarrow (T(G), f_{G,P}(s)) = (T(G), f_{T(G),P}(s)) \approx (G, f_{G,P}(s))$$

Key Property 2

- If $s_1 \neq s_i$, $T_{1,i}$ make the distribution ***looks random***

$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Transformation

Transformation T on a = 1 and b = 2

$$G = \begin{matrix} S_1 = (2,3,4) \\ S_2 = (1,3,5) \\ S_3 = (1,2,2) \\ S_4 = (5,3,4) \end{matrix} \xrightarrow{\text{T on a = 1 and b = 2}} T_{a,b}(G) = \begin{matrix} S'_1 = (1,3,4) \\ S'_2 = (2,3,5) \\ S'_3 = (1,2,1) \\ S'_4 = (5,3,4) \end{matrix}$$

- All values unchanged except for a and b
- **Value a and Value b swap with probability half**

Example: $P(s_1, s_i, s_5, s_3) \neq P(s_1, s_1, s_5, s_3)$

Output might not be same

$$f_{G,P}(s) \neq f_{T_{1,i}(G),P}(s)$$

$\Rightarrow$

Graph and Output less coupled

$T_{1,i}(G)$ Less coupled with $f_{G,P}(s)$

Use hybrid argument to argue distinguishing difference

Hybrid Distributions

Define hybrid where H_i is the distribution after applying i Transformations

- Each with randomly selected a, b

$H_0 = (G, f_{G,P}(s))$ Planted (0 transformation)

$H_1 = (T_{a_1,b_1}(G), f_{G,P}(s))$ Less Planted (1 transformation)

...

$H_i = (T_{a_i,b_i} \circ \cdots \circ T_{a_1,b_1}(G), f_{G,P}(s))$ Middle (i transformation)

...

$H_t = (T_{a_t,b_t} \circ \cdots \circ T_{a_1,b_1}(G), f_{G,P}(s))$ Random (t transformation)

How many transformations to get the instance look statistically close to random?

$$t = O(n \log(nm))$$

Search Algorithm

$H_0 \approx Planted \quad H_t \approx Random$

$H_i = (T_{a_i,b_i} \circ \cdots \circ T_{a_1,b_1}(G), f_{G,P}(s))$

Search Algorithm

What is s_i ?

↓

Apply r **Transformations** $T_{a,b}$ with random a, b to get H_r

↓

Apply $T_{1,i}$ to H_r

If $s_1 = s_i$ → Instance stays H_r

If $s_1 \neq s_i$ → Instance becomes like H_{r+1}

Hybrid Argument:

There exist a r such that Decision Oracle distinguishes H_r and H_{r+1}

Randomizing

Goal: After t transformation,

$$(T(G), f_{G,P}(s)) \approx (G, \text{random})$$

Randomizing Graph

After $t = O(n \log(nm/\varepsilon))$ transformation, hypergraph is independent of G

Transformation

Transformation T on a = 1 and b = 2

$$G = \begin{matrix} S_1 = (2,3,4) \\ S_2 = (1,3,5) \\ S_3 = (1,2,2) \\ S_4 = (5,3,4) \end{matrix} \xrightarrow{\text{T on a = 1 and b = 2}} T_{a,b}(G) = \begin{matrix} S'_1 = (1,3,4) \\ S'_2 = (2,3,5) \\ S'_3 = (1,2,1) \\ S'_4 = (5,3,4) \end{matrix}$$

- All values unchanged except for a and b
- **Value a and Value b swap with probability half**

$$T_{a_t,b_t} \circ \cdots \circ T_{a_1,b_1}(G) \approx_\varepsilon \text{Uniform}(G)$$

Proof Intuition

- Sample a_i, b_i randomly
- All values touched by transformation w.h.p

$$(T_{a_t,b_t} \circ \cdots \circ T_{a_1,b_1}(G), f_{G,P}(s)) \approx_\varepsilon (G', f_{G,P}(s))$$

Is this random?

Randomizing

Problem $(T_{a_t,b_t} \circ \cdots \circ T_{a_1,b_1}(G), f_{G,P}(s)) \approx_\varepsilon (G', f_{G,P}(s))$

Is this random?

Lemma: if G′,G is independently sampled, and s is random

$$(G', f_{G,P}(s)) \approx_\varepsilon (G, random)$$

Proof Intuition

- No information about G is leaked after knowing G′
- Output bits depends only on s. Fortunately, s is balanced w.h.p

Therefore, after $t = O(n \log(nm/\varepsilon))$ transformations with random ai,bi

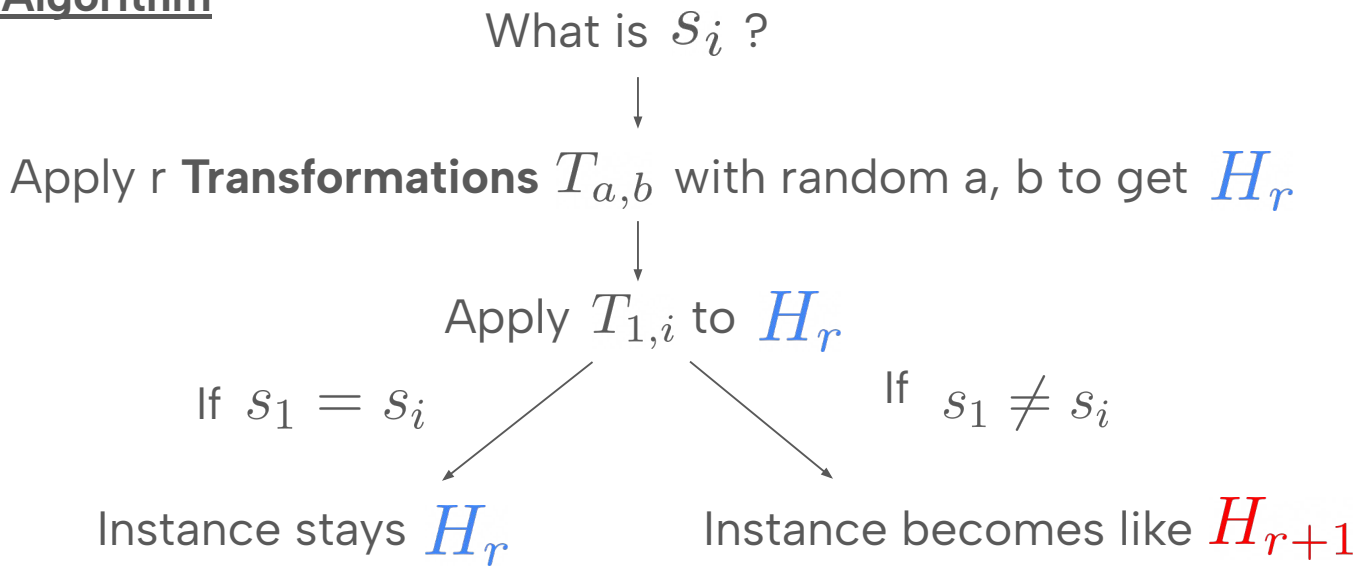
$$(T_{a_t,b_t} \circ \cdots \circ T_{a_1,b_1}(G), f_{G,P}(s)) \approx_\varepsilon (G', \text{random})$$

Final step

$H_0 \approx Planted \quad H_t \approx Random$

$H_i = (T_{a_i,b_i} \circ \cdots \circ T_{a_1,b_1}(G), f_{G,P}(s))$

Search Algorithm



Use Decision Oracle to distinguish these two cases
Amplify prediction with repetition

Result Summary

Decision algorithm with advantage ε and m samples

↓ Construct

Search Algorithm with with advantage $\Omega(\varepsilon)$ and M samples

	[App12]	[BRT25]	[TV26]
Sensitive	$M = O(m^3/\varepsilon^2), m = poly(n)$	$M = O(mn/\varepsilon^2), m = poly(n)$	$M = \tilde{O}(mn^2/\varepsilon^2)$ $m = poly(n)$
General	$M = O(n), m = O(n)$	x	

Open Problems

- Remove the additional n factor in [TV26] and match [BRT25]

Thank You

Generalization

Non-Constant Sparsity

Non constant sparsity d = polylog(n)

Minor additional requirement on m and eps

Distinct values in the hyperedges

Distinct values in each hyperedge

Our reduction still work with a slight loss in constant factor

Noisy Predicate

Motivated by LPN problem, reduction still applies with noise

Concluding Remarks

Implications

- Opens up the possibility of PRG from more predicates
- Lack of sensitivity means less structure, could be harder for attacks

Open Problems

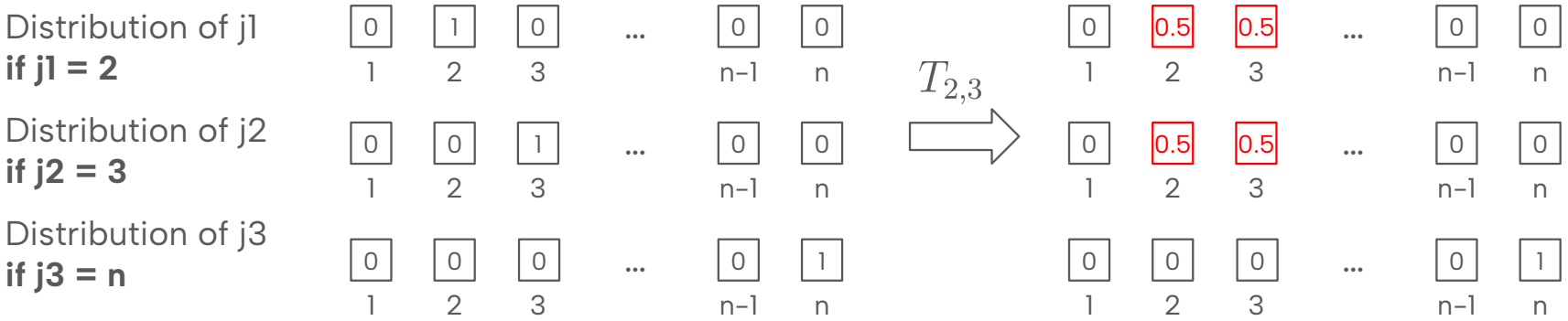
- Further lessen the gap of the output size between search and decision
- Utilize the reduction technique on other problems
- Find alternative family of Predicate and show OWF hardness

Proof Idea for Graph Randomisation

Describe the distribution of each value in hyperedge as n boxes

Value of ***i**–th* box describe the prob to get value ***i***

Initially, for each value only one box has value 1, the rest is all 0



After t random transformations, each box to have prob close to 1/n