

بسم الله الرحمن الرحيم

# جلسه پنجم

خلاصه سازی برای مهداده

## مرور: مساله تعداد متفاوت‌ها (F0)

---

• ورودی:

$$i_1, \dots, i_m \in [n] \quad \bullet$$

• خروجی:

• تعداد متفاوت‌ها

# خانواده الگوریتم :FM

## Algorithm FM:

1. Pick random hash function  $h : [n] \rightarrow [0, 1]$ .
2. Maintain in memory the smallest hash we've seen so far:  $X = \min_{i \in \text{stream}} h(i)$ .
3. `query()`: output  $1/X - 1$ .

KMV

## Algorithm FM+:

1. Instantiate  $s = \lceil 1/(\epsilon^2 \delta) \rceil$  FMs independently,  $\text{FM}_1, \dots, \text{FM}_s$ .
2. Let  $X_i$  be the output of  $\text{FM}_i$ .
3. Upon a query, output  $1/Z - 1$ , where  $Z = \frac{1}{s} \sum_i X_i$ .

$$P[|t - \tilde{t}| > \epsilon t] < \delta$$

$$\Theta(\epsilon^{-2} \ln \frac{1}{\delta})$$
 حافظه

## Algorithm FM++:

1. Instantiate  $q = \lceil 18 \ln(1/\delta) \rceil$  independent copies of FM+ with  $\eta = 1/3$ .
2. Output the median  $\hat{t}$  of  $\{1/Z_j - 1\}_{j=1}^q$  where  $Z_j$  is the output of the  $j$ th copy of FM+.

الگوریتم  
KMV



## استقلال K- طرفه

---

**$k$ -wise independent hash families.**

**Definition 2.2.6.** A family  $\mathcal{H}$  of functions mapping  $[a]$  to  $[b]$  is  $k$ -wise independent if  $\forall j_1, \dots, j_k \in [b]$  and  $\forall$  distinct  $i_1, \dots, i_k \in [a]$ ,

$$\mathbb{P}_{h \in \mathcal{H}}(h(i_1) = j_1 \wedge \dots \wedge h(i_k) = j_k) = 1/b^k$$

---

***k*-wise independent hash families.**

**Definition 2.2.6.** A family  $\mathcal{H}$  of functions mapping  $[a]$  to  $[b]$  is *k*-wise independent if  $\forall j_1, \dots, j_k \in [b]$  and  $\forall$  distinct  $i_1, \dots, i_k \in [a]$ ,

$$\mathbb{P}_{h \in \mathcal{H}}(h(i_1) = j_1 \wedge \dots \wedge h(i_k) = j_k) = 1/b^k$$

**Example.** The set  $\mathcal{H}$  of all functions  $[a] \rightarrow [b]$  is *k*-wise independent for every *k*.  $|\mathcal{H}| = b^a$  so  $h \in \mathcal{H}$  is representable in  $a \lg b$  bits.

*k*-wise independent hash families.

**Definition 2.2.6.** A family  $\mathcal{H}$  of functions mapping  $[a]$  to  $[b]$  is *k*-wise independent if  $\forall j_1, \dots, j_k \in [b]$  and  $\forall$  distinct  $i_1, \dots, i_k \in [a]$ ,

$$\mathbb{P}_{h \in \mathcal{H}}(h(i_1) = j_1 \wedge \dots \wedge h(i_k) = j_k) = 1/b^k$$

**Example.** The set  $\mathcal{H}$  of all functions  $[a] \rightarrow [b]$  is *k*-wise independent for every *k*.  $|\mathcal{H}| = b^a$  so  $h \in \mathcal{H}$  is representable in  $a \lg b$  bits.

**Example.** Let  $a = b = q$  for  $q = p^r$  a prime power and define  $\mathcal{H}_{poly}$  to be the set of all degree  $\leq k - 1$  polynomials with coefficients in  $\mathbb{F}_q$ , the finite field of order  $q$ .  $|\mathcal{H}_{poly}| = q^k$  so  $h \in \mathcal{H}$  is representable in  $k \lg p = k \lg a$  bits.

**Claim 2.2.7.**  $\mathcal{H}_{poly}$  is *k*-wise independent.

اثبات

- تنها یک چندجمله‌ای از این نقاط می‌گذرد

# الگوریتم KMV

- انتخاب تابع درهمساز  $h : [n] \rightarrow [M]$
- برای  $M = n^3$
- و  $h$  استقلال ۲ - طرفه
- به روزرسانی:  $k$  کوچکترین از مقدارهای  $(h_{i,j})$  را نگه دار
- $k = \lceil 24/\epsilon^2 \rceil$
- جواب:  $\tilde{t} = kM/X$
- ایده: همان روش قبلی (FM+)
- ۱) گستته شده.
- ۲) به جای کوچکترین،  $k$ -امین کوچکترین

• هدف: با احتمال خوب:  $(1 - \varepsilon)t \leq \tilde{t} \leq (1 + \varepsilon)t$ .

## تحليل الگوریتم

- حالت بد اول:  $\tilde{t} > (1 + \epsilon)t$

- حالت بد دوم:  $\tilde{t} < (1 - \epsilon)t$

-  $<=$  احتمال بد بودن = احتمال اول + احتمال دوم

• هدف: با احتمال خوب:  $(1 - \varepsilon)t \leq \tilde{t} \leq (1 + \varepsilon)t$ .

## تحليل الگوریتم

- حالت بد اول:  $\tilde{t} > (1 + \epsilon)t$

$$k \text{ تا درهم شده متمایز کوچک باشد} \quad \Leftrightarrow \quad \frac{kM}{(1 + \epsilon)t} > X \quad \Leftrightarrow \quad \tilde{t} = kM/X > (1 + \epsilon)t$$

- امین عدد متمایز کوچک باشد  $i: Y_i$

$$\text{احتمال حالت بد اول} = P \left[ \sum_i Y_i \geq k \right]$$

• هدف: با احتمال خوب:  $(1 - \varepsilon)t \leq \tilde{t} \leq (1 + \varepsilon)t$ .

## تحليل الگوریتم

- حالت بد اول:  $\tilde{t} > (1 + \epsilon)t$

$$\frac{kM}{(1 + \epsilon)t} \text{ امین عدد متمایز کوچکتر از } Y_i$$

$$EY < \frac{k}{(1 + \epsilon)} \quad \leftarrow \quad EY_i < \frac{k}{(1 + \epsilon)t}$$

$$E[Y^2] = E\left[\sum_i Y_i^2\right] = E\left[\sum_i Y_i\right] < \frac{k}{1 + e}$$

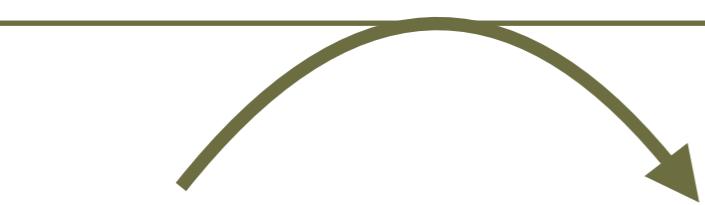
$$Var[Y] = \frac{k}{1 + e} - \left(\frac{k}{1 + e}\right)^2 < \frac{k}{1 + e}$$

۱ - از دو طرف  $EY$  کم کنیم

۲ - قدر مطلق بگیریم

$$P[|Y - EY| \geq k - EY] \text{ پس داریم}$$

جایگزینی  $EY < k/(1 + \epsilon)$



$$\mathbb{P}(Y \geq k) \leq \mathbb{P}(|Y - \mathbb{E} Y| > k(1 - 1/(1 + \varepsilon))) < \frac{k}{1 + \varepsilon} \cdot \frac{(1 + \varepsilon)^2}{k^2 \varepsilon^2} = \frac{1 + \varepsilon}{\varepsilon^2 k} < 1/6.$$

$$\frac{k\varepsilon}{1 + \epsilon}$$

$$EY < \frac{k}{(1 + \epsilon)}$$

$$Var[Y] < \frac{k}{1 + e}$$

بزرگتر از  $EY$

$$k = \lceil 24/\epsilon^2 \rceil$$

پس: احتمال حالت بد اول  $> 1/6$

$$(1 - \varepsilon)t \leq \tilde{t} \leq (1 + \varepsilon)t.$$

هدف ◉

## تحليل الگوریتم

- حالت بد دوم:

$$\text{کمتر از } k \text{ تا درهم شده متمایز کوچک} \quad \leq \quad \frac{kM}{(1 - \varepsilon)t} < X \quad \leq \quad \tilde{t} = kM/X < (1 - \varepsilon)t \quad -$$

$$Z = \sum Z_i \quad -$$

$\frac{kM}{(1 - \varepsilon)t}$  : درهم شده متمایز  $i$ -ام کوچکتر مساوی

- احتمال حالت بد دوم =

$$(1 - \varepsilon)t \leq \tilde{t} \leq (1 + \varepsilon)t.$$

هدف ◉

$P[Z < k] =$  احتمال حالت بد دوم

- حالت بد دوم:  $(1 - \epsilon)t > \tilde{t}$

-  $Z_i$ : درهم شده متمایز - ام کوچکتر مساوی  $\frac{kM}{(1 - \epsilon)t}$

$$\frac{k}{(1 - \epsilon)t} - \frac{1}{M} < EZ_i \leq \frac{k}{(1 - \epsilon)t} \quad \text{پس}$$

$$k\left(1 + \frac{3}{4}\epsilon\right) < \frac{k}{1 - \epsilon} - \frac{\epsilon k}{4} < \frac{k}{1 - \epsilon} - \frac{t}{M} < EZ \leq \frac{k}{1 - \epsilon}$$

1 >  $t/n$   
 $\epsilon > 1/n^2$   
 $k/4 > 1$

$$k(1 + \frac{3}{4}\epsilon) < EZ \leq \frac{k}{1 - \epsilon}$$

$$Var[Z] = \sum E[Z_i^2] - (E[Z])^2 \leq E[Z]$$

$$\begin{aligned} P[Z < k] &= P[EZ - Z > EZ - k] \leq P[|EZ - Z| > EZ - k] \\ &\leq P[|EZ - Z| > k(1 + \frac{3}{4}\epsilon) - k] \\ &< E[Z] \cdot (\frac{1}{3/4\epsilon k})^2 \\ &< \frac{k}{1 - \epsilon} \cdot \frac{16}{9\epsilon^2 k^2} < \frac{16}{9(1 - \epsilon)} \cdot \frac{1}{\epsilon^2 k} < 1/6, \end{aligned}$$

$$1 - \epsilon > 1/2$$

$$k = \lceil 24/\epsilon^2 \rceil$$

# جمع‌بندی :KMV

- انتخاب تابع درهم‌ساز  $h : [n] \rightarrow [M]$
- برای  $M = n^3$
- و  $h$  استقلال ۲- طرفه
- به روزرسانی:  $k$  کوچکترین از مقدارهای  $(j, h(i_j))$  را نگه دار
- $k = \lceil 24/\epsilon^2 \rceil$
- جواب:  $\tilde{t} = kM/X$
- احتمال خطأ >  $1/3$

FM++

جايگزيني  
با  
KMV

1. Instantiate  $q = \lceil 18 \ln(1/\delta) \rceil$  independent copies of FM+ with  $\eta = 1/3$ .
2. Output the median  $\hat{t}$  of  $\{1/Z_j - 1\}_{j=1}^q$  where  $Z_j$  is the output of the  $j$ th copy of FM+.

# بهترین الگوریتم‌ها تا کنون

خطای ثابت: حافظه  $O(1/\epsilon^2 + \log n)$

- برای خطای  $\delta$ :  
 $\Theta(\epsilon^{-2} \log(1/\delta) + \log n)$
- الگوریتم HyperLogLog
  - ضریب حافظه  $\lg \lg n$  بیشتر
  - فرض: تابع  $h$  تصادفی واقعی است
  - اما خیلی سریع تر

# روش دیگر: نمونه‌گیری هندسی

$$h : [n] \rightarrow \{0, \dots, \log n\}$$

2-wise independent

$$\mathbb{P}(h(i) = j) = 2^{-(j+1)} \text{ for } j < \log n$$

$$\mathbb{P}(h(i) = \log n) = 1/n$$

برای یک عدد  $k$ :

اگر تعداد اعداد متمایز کمتر مساوی  $k$ : پاسخ صحیح

: یک ساختمان داده که:  $D_r$

اگر نه: مهم نیست (مثلا ۱

# مرحله ۱: الگوریتم با ضریب تقریب ثابت

جواب

۱: بزرگترین  $r$  که  $D_r$  خالی نیست.

جواب =  $2^l$

به روزرسانی

$D_0$   
 $D_1$



$D \log n$

برای  $D$  ها داریم  $k=1$

# مرحلة ١: الگوريتم با ضريرب تقرير ثابت - تحليل

$$L := \log_2 t.$$

**Lemma 2.2.8.**  $\mathbb{P}(\ell > L + 4) \leq 1/8$ .

$$\mathbb{E}_h F_0(r) = t/2^r.$$

Dr : تعداد متمايز در  $F_0(r)$

$$\begin{aligned}\mathbb{P}(\ell > L + 4) &= \mathbb{P}(\exists r > L + 4 : F_0(r) > 0) \\ &\leq \sum_{r=\lceil L+4 \rceil}^{\log n} \mathbb{P}(F_0(r) > 0) \text{ (union bound)} \\ &= \sum_{r=\lceil L+4 \rceil}^{\log n} \mathbb{P}(F_0(r) \geq 1) \text{ ( $F_0(r)$  is an integer)} \\ &= \sum_{r=\lceil L+4 \rceil}^{\log n} \mathbb{P}\left(F_0(r) \geq \frac{2^r}{t} \cdot \mathbb{E} F_0(r)\right) \leq \sum_{r=\lceil L+4 \rceil}^{\log n} \frac{t}{2^r} < \sum_{q=4}^{\infty} 2^{-q} = \frac{1}{8}.\end{aligned}$$

ماركوف

**Lemma 2.2.9.**  $\mathbb{P}(F_0(\lceil L - 4 \rceil) = 0) < 1/8$ .

$$q = F_0(\lceil L - 4 \rceil)$$

$$\mathbb{E} q = t/2^{\lceil L-4 \rceil}$$

$$Var[q] < \mathbb{E} q$$

متغیرهای برنولی مستقل

$$\begin{aligned}\mathbb{P}(q = 0) &\leq \mathbb{P}(|q - \mathbb{E} q| \geq \mathbb{E} q) \\ &< \mathbb{P}(|q - \mathbb{E} q| \geq \sqrt{\mathbb{E} q} \sqrt{Var[q]}) \\ &\leq \frac{1}{\mathbb{E} q} \text{ (Chebyshev's inequality),} \\ &\leq \frac{1}{8}\end{aligned}$$

→ **Lemma 2.2.10.**  $2^\ell \in [\frac{1}{16}t, 16t]$  with probability at least  $3/4$ .

# مرحله ۲: تقریب ۱ EPSILON+

$$k_2 = \lceil 512/\varepsilon^2 + 68/\varepsilon \rceil.$$

$t > k_2$  در غیر این صورت حل است!

جواب

: پاسخ روش تقریب با ضریب ثابت  $\hat{t}$

$$r^* = \lceil \log_2(\varepsilon^2 \hat{t} / 32) \rceil$$

$2^{r^*} \times F_0(r^*) =$  جواب

به روز رسانی

روش تقریب با ضریب ثابت

D0

D1

i

i

Dh(i)

D log n

برای Dها با ظرفیت  $k_2$

$$r^* = \lceil \log_2(\varepsilon^2 \hat{t} / 32) \rceil$$

با احتمال  $3/4$  داریم:

( Lemma 2.2.10.  $2^\ell \in [\frac{1}{16}t, 16t]$  with probability at least  $3/4$ . چون )

$$Var_h[F_0(r^*)] \leq t/2^{r^*},$$

$$P[|F_0 - EF_0| > x] < 1/x^2 Var[F_0] = 1/9 \quad x = 3\sqrt{Var[F_0]}$$

با احتمال  $8/9$  (به شرط قبلی) داریم:

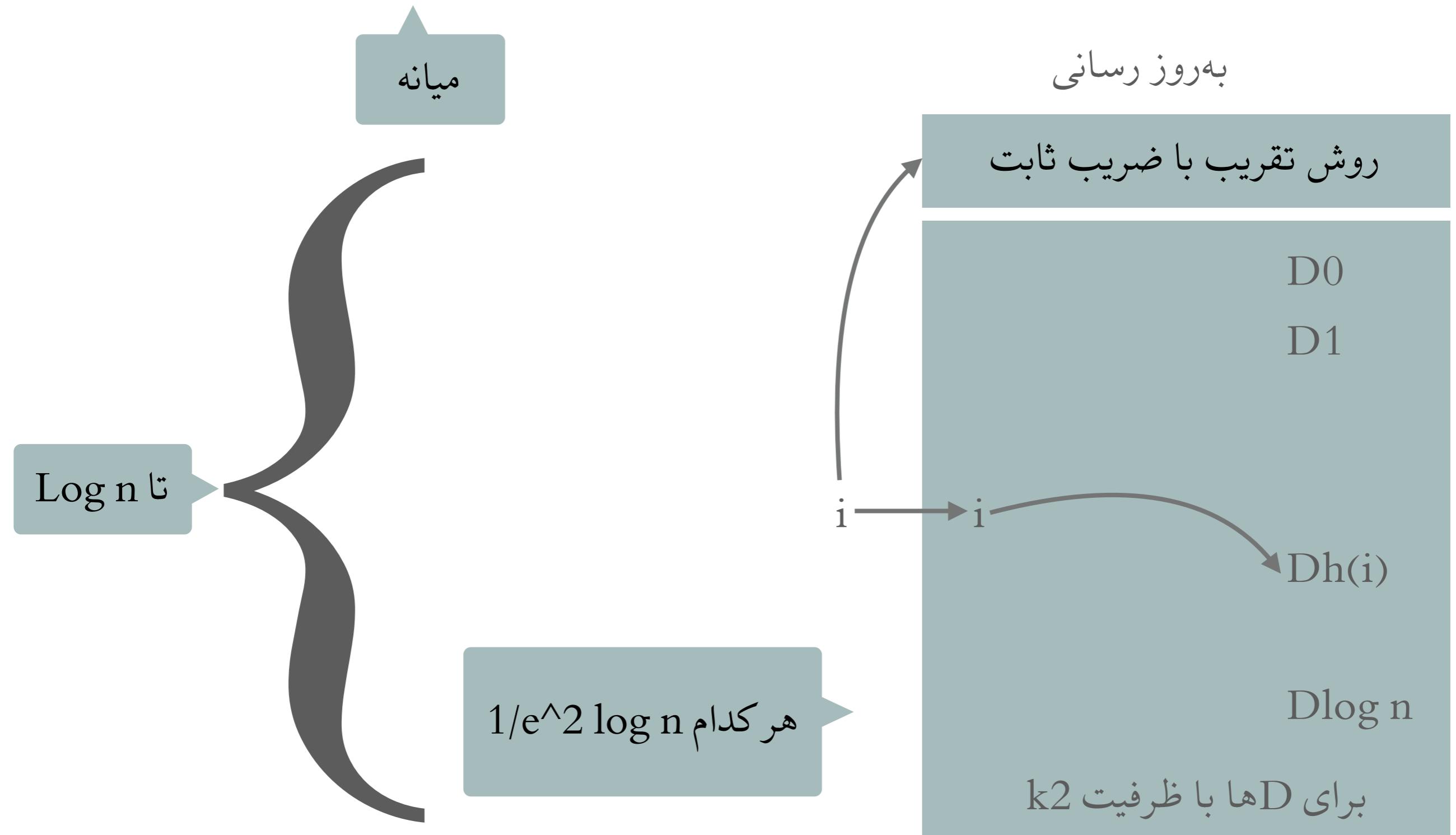
$$F_0 = EF_0 \pm x = t/2^{r^*} \pm 3\sqrt{t/2^{r^*}} = t/2^{r^*} \pm C/\epsilon = (1 \pm C'\epsilon)t/2^{r^*}$$

$$r^* = \lceil \log_2(\varepsilon^2 \hat{t} / 32) \rceil$$

$$1/\epsilon < \epsilon \cdot t/2^{r^*}$$

$$1/\varepsilon^2 < t/2^{r^*} \leq 512/\varepsilon^2$$

**Theorem 2.2.11.** There is an algorithm for  $(1 + \varepsilon)$ -approximation of  $F_0$  with probability  $1 - \delta$  using  $O(\varepsilon^{-2} \log^2 n \log(1/\delta))$  bits of memory.



# کران پائین





کران پایین برای  
تعداد اعداد  
متفاوت F0

**Theorem 3.1.1.** Suppose  $\mathcal{A}$  is a deterministic streaming algorithm that computes the number of distinct elements exactly. Then  $\mathcal{A}$  uses at least  $n$  bits of memory.

$$\text{Enc} : \{0, 1\}^n \rightarrow \{0, 1\}^S$$

دو ورودی را به یک خروجی نمی‌برد

$$S \geq n$$

ایده

$\text{Enc}(x)$ :

$$x \in \{0, 1\}^n$$

۱ - یک رشته با اعدادی که اندیشان در ورودی هست (مرتب)

۲ - رشته را به الگوریتم  $A$  بدهیم.

۳ - خروجی: حافظه  $A$

# تابع ENC یک‌به‌یک است:

Enc(x):

- ۱ - یک رشته با اعدادی که اندیشان در ورودی هست (مرتب)
- ۲ - رشته را به الگوریتم A بدهیم.
- ۳ - خروجی (M): حافظه A

Dec(M)

```
s ← A.query() // support size of x, i.e. |{i : xi ≠ 0}|  
x ← (0, 0, ..., 0)  
for i = 1 ... n:  
    A.update(i) // append i to the stream  
    r ← A.query() // will either be s or s + 1  
    if r = s: // Encoder must have included i, so it wasn't a new distinct element  
        xi ← 1  
    s ← r  
return x
```