Subject:
Date

الگوریتم های تصادفی

مرور بر احتمال:

$P(A|B) = \frac{P(A \cap B)}{P(B)}$ مستقل $A, B \leftrightarrow P(A \cap B) = P(A).P(B)$

مستقل $A_1, A_2, \dots, A_k \leftrightarrow \forall J \subseteq \{1, \dots, k\}: P(\bigcap_{i \in J} A_i) = \prod_{i \in J} P(A_i)$ ← این معادل مستقل بودن دو به دوی رویدادهاست؛ بلکه قوی‌تر از آن است.

مستقل Z و X متغیر تصادفی $\Pr(X=a \wedge Z=b) = \Pr(X=a).\Pr(Z=b) \leftrightarrow E(XZ) = E(X).E(Z)$

مسئله: نشان دهید تورنمنتی وجود دارد که حداقل $\frac{n!}{2^{n-1}}$ مسیر همیلتونی دارد. (تورنمنت گراف کاملی است که یال‌های آن جهت‌دار شده‌اند.)

تورنمنتی تصادفی تولید می‌کنیم. هر مسیر اویلری (از بین $n!$ مسیر همیلتونی ممکن) به احتمال $\frac{1}{2^{n-1}}$ در این تورنمنت وجود دارد. بنابراین اگر X تعداد مسیرهای همیلتونی گراف باشد:

$X = \sum_{\sigma \in \text{ممکن}} X_\sigma$ $\quad X_\sigma = \begin{cases} 1 & \sigma \text{ همیلتونی} \\ 0 & \text{در غیر این صورت} \end{cases}$ $\quad \sigma$ یک مسیر

$E(X) = \sum E(X_\sigma) = \frac{n!}{2^{n-1}}$ $\quad$ * استفاده از خاصیت خطی E

مسئله: G گرافی با m یال است. نشان دهید $H \subseteq G$ وجود دارد که دوبخشی است و حداقل $\frac{m}{2}$ یال دارد.

گراف تصادفی دوبخشی H را به این صورت می‌سازیم که هر رأس را به احتمال $\frac{1}{2}$ در بخش اول و به احتمال $\frac{1}{2}$ در بخش دوم قرار می‌دهیم و سپس یال‌های دوبخشی را اضافه می‌کنیم. هر یال G با احتمال $\frac{1}{2}$ در H موجود است، پس امید ریاضی تعداد یال‌های H، $\frac{m}{2}$ است.

مسئله: اگر $\sigma(G)$ بزرگترین مجموعه مستقل رئوس باشد و $d = \frac{2m}{n}$ میانگین درجات، نشان دهید: $\sigma(G) \geq \frac{n}{2d}$

ابتدا یک مجموعه S را به طور تصادفی انتخاب می‌کنیم و سپس برای حذف یال‌های بین آن رئوس، به همان تعداد رأس از مجموعه حذف می‌کنیم.

امید ریاضی

اگر تعداد رئوس باقیمانده $\frac{n}{2d}$ باشد مسئله حل است. مجموعه ابتدایی را به این صورت انتخاب می‌کنیم که هر رأس با احتمال p انتخاب می‌شود.

X: اندازه S $\quad Z$: تعداد یال‌های S $\quad E(X) = np \quad E(Z) = mp^2 \quad E(X-Z) = p(n-mp) \xrightarrow{p=\frac{1}{d}} \frac{n}{2d}$

P4PCO

Subject:
Date

قضیه مارکوف: (X نامنفی)

$$Pr(X \geq a) \leq \frac{\mathbb{E}(X)}{a}$$

$$\mathbb{E}(X) = \sum_{b \geq 0} b \cdot Pr(X=b) = \sum_{0 \leq b < a} b \cdot Pr(X=b) + \sum_{b \geq a} b \cdot Pr(X=b) \geq \sum_{b \geq a} a \cdot Pr(X=b) = a \cdot Pr(X \geq a)$$

$$Var(X) = \sum \left(X(a_i) - \mathbb{E}(X)\right)^2 \cdot Pr(a_i) \quad \text{(پیشامد)}$$

انحراف معیار: $\sigma(X) = \sqrt{Var(X)}$

قضیه چبیشف: ($\lambda\sigma$ ↑ a)

$$Pr\left(|X - \mathbb{E}(X)| \geq a\right) \leq \frac{Var(X)}{a^2} = \frac{1}{\lambda^2}$$

$$Var(X) = \mathbb{E}\left((X - \mathbb{E}(X))^2\right) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$$

$$Cov(X,Y) = \mathbb{E}\left((X - \mathbb{E}(X))(Y - \mathbb{E}(Y))\right) = \mathbb{E}(XY) - \mathbb{E}(X)\cdot\mathbb{E}(Y)$$

$$Var\left(\sum X_i\right) = \sum Var(X_i) + \sum Cov(X_i, X_j)$$

اثبات نامساوی چبیشف: (مارکوف)

$$Pr\left(|X - \mathbb{E}(X)| \geq a\right) = Pr\left((X - \mathbb{E}(X))^2 \geq a^2\right) \leq \frac{\mathbb{E}\left((X - \mathbb{E}(X))^2\right)}{a^2} = \frac{Var(X)}{a^2}$$

حال جواهیم با استفاده از نامساوی چبیشف، نامساوی روبرو را اثبات کنیم:

$$\binom{2m}{m} \geq \frac{2^{2m}}{4\sqrt{m}+2}$$

$$X = X_1 + X_2 + \dots + X_{2m} \qquad X_i = \begin{cases} 1 & \text{احتمال } \frac{1}{2} \\ 0 & \text{احتمال } \frac{1}{2} \end{cases} \qquad \mathbb{E}(X) = m \qquad Var(X) = \frac{m}{2}$$

$$Pr\left(|X - m| \geq \sqrt{m}\right) \leq \frac{\frac{m}{2}}{(\sqrt{m})^2} = \frac{1}{2} \Rightarrow Pr\left(|X - m| < \sqrt{m}\right) \geq \frac{1}{2}$$

$$\frac{1}{2} \leq Pr\left(|X - m| < \sqrt{m}\right) = \sum_{|k| < \sqrt{m}} \binom{2m}{m+k} \left(\frac{1}{2}\right)^{2m} \leq (2\sqrt{m}+1)\binom{2m}{m}\left(\frac{1}{2}\right)^{2m}$$

Subject:
Date

$G(n, \frac{1}{2})$: گراف تصادفی n رأسی که بین هر یال آن با احتمال $\frac{1}{2}$ وجود دارد.

$\mathbb{E}(\max \text{degree}(G(n,\frac{1}{2}))) = ?$

X_i : درجه رأس iام $\mathbb{E}(X_i) = \frac{n-1}{2}$ $X = \max(X_1, X_2, \dots, X_n)$

توجه کنید که $\mathbb{E}(\max(X_1,\dots,X_n))$ با $\max_{i=1}^{n} \mathbb{E}(X_i)$ برابر نیست؛ مثلاً اگر X برنولی باشد: $\mathbb{E}(\max(X, 1-X)) = 1 \neq \max(\mathbb{E}(X), \mathbb{E}(1-X))$

$Pr(X \geq a) = Pr(X_1 \geq a \vee X_2 \geq a \vee \dots \vee X_n \geq a) \leq \sum Pr(X_i \geq a)$

اگر بتوانیم طوری نشان دهیم $Pr(X_i \geq a)$ کوچکتر از $\frac{c}{n^2}$ است (یا $O(\frac{1}{n^2})$) آنگاه $Pr(X \geq a)$ از $O(\frac{1}{n})$ خواهد بود یا آنگاه:

$$\mathbb{E}(X) = \sum_{i=0}^{n-1} i \cdot Pr(X=i) \leq \sum_{i=0}^{a} i \cdot Pr(X=i) + \underbrace{a \cdot Pr(X \geq a)}_{O(1)} \leq a + O(1)$$

طبقت: $a \leq n$

قضیه چرنوف: اگر $X = X_1 + \dots + X_n$ که X_i ها مستقل هستند، آنگاه:

$$X_i = \begin{cases} 1 & \text{احتمال } \frac{1}{2} \\ -1 & \text{احتمال } \frac{1}{2} \end{cases}$$

$$Pr(X \geq t) \leq e^{\frac{-t^2}{2n}}, \quad Pr(X \leq -t) \leq e^{-\frac{t^2}{2n}}$$

اثبات: با انتخاب u مناسب (با مشتق گیری) به نامساوی چرنوف می‌رسیم.

$Z = e^{uX}$ ثبت z : $Pr(X \geq t) = Pr(Z \geq e^{ut}) \leq \frac{\mathbb{E}(Z)}{e^{ut}} = \frac{\prod \mathbb{E}(e^{uX_i})}{e^{ut}} = \frac{\left(\frac{e^u + e^{-u}}{2}\right)^n}{e^{ut}}$

نسخه دیگری از چرنوف: اگر X_i ها برنولی با پارامتر p_i باشند، برای هر $\delta > 0$: $\mu = \sum p_i$, $Pr(X \geq (1+\delta)\mu) \leq \left(\frac{e^{1+\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$

با استفاده از نسخه بالا از چرنوف a مناسب که احتمالاً به صورت $\frac{n-1}{2} + c\sqrt{n \lg n}$ را پیدا می‌کنیم. (برای مثال استدلال ریاضی ماکزیمم درجه گراف تصادفی)

Subject:
Date

discrepency: $X = \{a_1, a_2, \dots, a_n\}$, $F \subset 2^X$, $|F| = m$, $\chi : X \to \{-1, 1\}$

F مجموعه‌ای m عضوی از زیرمجموعه‌های X است. χ یک رنگ‌آمیزی با دو رنگ روی اعضای X است discripency برای F نسبت به رنگ‌آمیزی χ، ماکزیمم اختلاف رنگ در بین اعضای F است. اختلاف رنگ یعنی اختلاف تعداد دو رنگ استفاده شده در رنگ‌آمیزی یک زیرمجموعه از X.

$$disc(F, \chi) = \max_{S \in F} \left| \sum_{a \in S} \chi(a) \right| \qquad disc(F) = \min_{\chi} disc(F, \chi)$$

discrepency برای F، discrepency برای F نسبت به بهترین رنگ‌آمیزی ممکن است (در واقع نسبت به بهترین رنگ‌آمیزی)

قضیه: $disc(F) \le \sqrt{2n \cdot \lg(2m)}$ و اگر δ ماکزیمم اندازه زیرمجموعه‌های موجود در F باشد: $disc(F) \le \sqrt{2\delta \cdot \ln(2m)}$

اثبات: رنگ‌آمیزی را به صورت تصادفی انجام می‌دهیم. با استفاده از قضیه چرنوف داریم:

$$Pr(|\chi(S)| \ge t) < 2e^{\frac{-t^2}{2|S|}}$$

$$t = \sqrt{2|S| \ln(2m)} : \quad Pr(|\chi(S)| \ge t) \le 2e^{-\ln(2m)} = 2\frac{1}{2m} = \frac{1}{m}$$

$$F = \{S_1, S_2, \dots, S_m\} \quad Pr(\chi(S_1) \ge t \vee \chi(S_2) \ge t \vee \dots \vee \chi(S_m) \ge t) \le \sum Pr(\chi(S_i) \ge t) < m \times \frac{1}{m} = 1$$

بنابراین احتمال این که اختلاف هیچ زیرمجموعه‌ای کمتر از t باشد بیشتر از صفر است؛ بنابراین این حالت رخ می‌دهد؛ یعنی در بین همه رنگ‌آمیزی‌ها ممکن رنگ‌آمیزی χ^* وجود دارد که $disc(F, \chi^*) \le t$؛ بنابراین: $disc(F) \le t$ ■

Subject:
Date

الگوریتم محاسبه میانه: Median:

اگر S مجموعه‌ای شامل n عدد باشد:

$x \in S : rank(x) = |\{y \mid y < x\}|$

δ-median: $(\frac{1}{2} - \delta)n \leq rank(x) \leq (\frac{1}{2} + \delta)n$

موفقیت کاردلو (به دست آمدن جواب قطعی نیست)

```
Approx Median 1 (S, A)
n ← random (1 ... n)
x* ← A[n]
for i=1 to n do
    if A[i] < x* k ← k+1
if (1/2 - δ)n ≤ k ≤ (1/2 + δ)n return x*
else return error
```

زمان اجرا: $O(n)$ احتمال موفقیت: $1 - (1-2\delta)^c$

```
Approx Median 2 (S, A)
j ← 1
repeat result ← Approx Median 1 (S, A) ; j ← j+1
until result ≠ error or j ≠ c+1
return result
```

زمان اجرا: $O(n)$ احتمال موفقیت: 2δ

چون احتمال موفقیت در هر بار فراخوانی 2δ است، امید ریاضی دفعات تکرار تا رسیدن به موفقیت $\frac{1}{2\delta}$ است پس زمان اجرا $O(\frac{n}{2\delta})$ است.

لاس وگاس (زمان اجرا تصادفی است)

```
Approx Median 3 (S, A)
repeat result ← Approx Median 1 (S, A)
until result ≠ error
return result
```

Subject:
Date

Hiring: مسئله استخدام:

آژانسی n نفر را برای استخدام به شرکتی معرفی می‌کند. شرکت هر روز با یک نفر می‌تواند مصاحبه کند. شرکت ابتدا نفر اول را استخدام می‌کند. با هر مصاحبه اگر آدم بهتری پیدا شد، نفر قبلی اخراج می‌شود و نفر جدید جایگزین می‌شود. اما هر اخراج برای شرکت به اندازه f هزینه دارد. شرکت چطور می‌تواند با کمترین هزینه این عملیات را در n روز اجرا کند تا در پایان بهترین فرد را داشته باشد؟ (آژانس می‌تواند بدعنقی کند و لیست را به صورت صعودی مرتب کند)

برای جلوگیری از بدعنقی آژانس، شرکت لیست را به صورت تصادفی به هم می‌زند و با آن ترتیب مصاحبه را انجام می‌دهد. امید ریاضی هزینه چقدر است؟

احتمال این که نفر i‌ام از نفرات قبلی بهتر باشد $\frac{1}{i}$ و بنابراین مجبور شویم در مرحله i‌ام هزینه f را بدهیم. برابر است با $\frac{1}{i}$؛ زیرا نفرات اول تا i‌ام در یک جایگشت تصادفی هستند.

$$\sum_{i=1}^{n} \frac{1}{i} f = f \cdot \lg n$$

تولید جایگشت تصادفی:

```
for i=1 to n
    x ← random (i .. n)
    swap (A(n), A(i))
```

این الگوریتم تولید می‌کند n^n جایگشت ممکن را تولید می‌کند اما اگر به جای i قرار دهیم یک، یکنواخت نخواهد بود؛ زیرا n^n به $n!$ بخش‌پذیر نیست.

Subject:
Date

مسئله انتخاب: می‌خواهیم iامین عنصر مجموعه‌ای از اعداد (از نظر مقدار) بیابیم.

```
Select (S, i)
    if |S|=1 then return the only elemet
    else pick a pivot x_piv ∈ S
        S< ← {x∈S: x<x_piv}, S> ← {x∈S: x>x_piv}, k ← |S<|
        if k=i-1 return x_piv
        else if k>i-1 then return Select (S<, i)
        else return Select (S>, i-k-1)
```

$$T_{exp}(n) = O(n) + \sum_{1 \le j \le n} Pr(\text{rank } x_{piv} = j)\, T_{exp}(\max\{j, n-j\})$$

می‌توان رابطه بالا را به صورت ساده‌تر روبرو نوشت: (رنک x_{piv} به احتمال $\frac{1}{2}$ بین $\frac{n}{4}$ و $\frac{3n}{4}$ است)

$$T(n) = O(n) + \frac{1}{2}T\left(\frac{3}{4}n\right) + \frac{1}{2}T(n-1) \xrightarrow[\text{حدس می‌زنیم } O(n)]{\text{روش حدس و استقرا}} T(n) = O(n)$$

اثبات دقیق روی صفحه درس است و ثابت می‌کنیم

مرتب‌سازی:

Randomized quick sort:

مرتب‌سازی سریع شبیه Select است با این تفاوت که در هر مرحله هم $S_<$ فراخوانی می‌شود و هم $S_>$ و آرایه را مرتب می‌کند.

$$X_{ij} = \begin{cases} 1 & \text{اگر } x_i \text{ با } x_j \text{ مقایسه شود} \\ 0 & \text{o.w.} \end{cases}$$

اگر درخت بازگشتی را رسم کنیم، مجموعه $\{x_i, x_{i+1}, \dots, x_j\}$ همیشه در کنار هم خواهند بود تا جایی که یکی از این $j-i+1$ تا (به عنوان pivot) انتخاب شود.

اگر pivot یکی از x_i و x_j باشد، این دو عدد با هم مقایسه می‌شوند و اگر عضو دیگری باشد، x_i و x_j هیچ وقت مقایسه نمی‌شوند.

$$\mathbb{E}(X) = \sum\sum \mathbb{E}(X_{ij}) = \sum\sum Pr(X_{ij}=1) = \sum\sum \frac{2}{j-i+1} = 2\left\{1+\frac{1}{2}+\frac{1}{3}+\dots\right. = O(n \lg n)$$

Subject:
Date

راه دیگر این است که درخت بازگشتی را بررسی کنیم. در الگوریتم، اجرای بر سطح از درخت دقیقاً $O(n)$ زمان می‌برد. کافیست امید ریاضی ارتفاع درخت را بررسی کنیم. (توجه کنید که ارتفاع درخت برابر با ارتفاع بلندترین برگ آن است.)

اگر X_1 و ... و X_n برگ‌های درخت باشند، می‌خواهیم $\mathbb{E}[\max\{X_1,\dots,X_n\}]$ را بیابیم. توجه کنید که: $\mathbb{E}(X_i) = O(\lg n)$ (چرا؟)

اگر بتوانیم نشان دهیم $P(X \geq 4\lg n) = \Pr(X_1 \geq 4\lg n, \dots, X_n \geq 4\lg n) \leq \sum \Pr(X_i \geq 4\lg n)$ آنگاه $\Pr(X_i \geq 4\lg n) \leq \frac{1}{n^2}$

$\leq \frac{1}{n}$

در این صورت خواهیم داشت: $\mathbb{E}(X = \max(X_1,\dots,X_n)) \leq \sum_{i=1}^{4\lg n} \Pr(X=i)\times i + \overbrace{\sum_{i=4\lg n}^{n} \Pr(X=i)\times n}^{O(1)} = O(\lg n)$

مستند بیچ و مهره: راه‌حل deterministic در مقاله‌ای در ژورنال رورو: SIAM Journal on descrete mathematics

دو جلسه نوشتم!

Subject:
Date

قضیه Yao Min Max :

فرض کنید A الگوریتم تصادفی است (در واقع مجموعه‌ای از الگوریتم‌های قطعی که روی آنها توزیع احتمالاتی داریم) و X مجموعه ورودی‌های ممکن برای الگوریتم است (که روی این ورودی‌ها ممکن نیز توزیع احتمالاتی داریم) و تابع c هزینه یک الگوریتم را به ازای یک ورودی خاص مشخص می‌کند: (مثلاً تعداد مقایسه‌ها در یک الگوریتم مرتب‌سازی)

$$\max_{x \in X} \mathbb{E}(c(A, x)) \geq \min_{a \in A} \mathbb{E}(c(a, X))$$

اثبات:

$$A = \{A_1, \dots, A_n\} \quad (p_1 \dots p_n) \qquad X = \{x_1, \dots, x_m\} \quad (q_1 \dots q_m)$$

$$\max_{x \in X} \mathbb{E}(c(A, x)) = H \qquad \min_{a \in A} \mathbb{E}(c(a, X)) = F$$

$$H \geq \sum_{i=1}^{n} p_i\, c(A_i, x_j) \geq \sum_{j=1}^{m} q_j \sum_{i=1}^{n} p_i\, c(A_i, x_j) = \sum_{i=1}^{n} p_i \sum_{j=1}^{m} q_j\, c(A_i, x_j) \geq \sum_{i=1}^{n} p_i\, F = F$$

نتیجه: هیچ الگوریتم تصادفی در زمان بهتر از $n \lg n$ مرتب‌سازی را انجام نمی‌دهد، چون هیچ الگوریتم قطعی وجود ندارد و طبق قضیه بالا.

(توجه کنید که زمان یک الگوریتم تصادفی، همان مقدار میانگین حساب در قضیه بالا است.)

مسئله: تعدادی نقطه روی خط داریم. می‌خواهیم تعدادی بازه به طول یک انتخاب کنیم که همه این نقاط را پوشش دهد. Unit Clustering

نسخه آنلاین: نقاط یک به یک ظاهر می‌شوند. در هر مرحله یا یک بازه اضافه می‌کنیم و نقطه جدید را به آن بازه نسبت می‌دهیم یا بازه جدیدی اضافه می‌کنیم و نقطه جدید را به یکی از بازه‌های قبلی نسبت می‌دهیم. توجه کنید که بازه‌ای که به نقاط نسبت داده نشده، تا انتها عوض می‌شوند (نمی‌توانیم بازه‌ها را عقب ببریم، بدون این که نقاط قبلی که پوشش داده‌اند، از پوشش آنها خارج شوند.)

Subject:
Date

در این مسئله، عدد رقابتی برای یک الگوریتم آنلاین عبارت است از تعداد دفعاتی که در آن الگوریتم تقسیم بر تعداد بازه‌های بهترین جواب آفلاین.

می‌خواهیم نشان دهیم می‌توان الگوریتم تصادفی با عدد رقابتی کمتر از $\frac{4}{3}$ ارائه داد. برای این کار مثالی از یک توزیع روی ورودی‌ها ارائه می‌دهیم.

می‌خواهیم $\min_{a \in A} \mathbb{E}(c(a,X))$ را برای X دوبرو بیابیم. توجه کنید که a یک الگوریتم قطعی است.

$X = \{ (1,2) \text{ با احتمال } p \ ,\ (1,2,3,0) \text{ با احتمال } 1-p \}$

$\mathbb{E}(c(a,X))$

الگوریتم‌های قطعی برای ورودی X به یکی از این صورت‌ها عمل می‌کنند:

① [1][2] , [0][1][2][3] $\quad \frac{2}{1}\times p + \frac{3}{2}(1-p)$

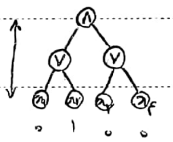
② [1][2] , [0][1][2][3] $\quad \frac{2}{1}\times p + \frac{2}{2}(1-p) \rightarrow \frac{3}{2} - \frac{p}{2}$

③ [1][2] , [0][1][2][3] $\quad \frac{2}{1}\times p + \frac{3}{2}(1-p) \rightarrow p+1$

و الگوریتم‌های بدتر از ②

برای این‌که $\min\{p+1, \frac{3}{2}-\frac{p}{2}\}$ بزرگترین مقدار ممکن باشد، قرار می‌دهیم $p = \frac{1}{3}$ و بنابراین $\min_{a \in A} \mathbb{E}(c(a,X))$ برابر با $\frac{4}{3}$ خواهد شد. ■

مسئله: می‌خواهیم تعداد دوبرو را می‌سه کنیم.



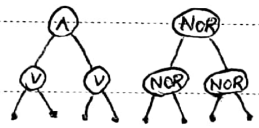
توجه کنید 2^{2k} عضو داریم.

یک الگوریتم قطعی این است که به 4^k عدد را بخواند و عبارت را می‌سه کند. می‌خواهیم الگوریتمی تصادفی ارائه دهیم که با خواندن قسمتی از ورودی‌ها جواب را می‌سه کند.

Subject :
Date

اگر بگیریم $n = 4^k$ ، الگوریتم ارائه شده $3^k = n^{0.749}$ است. می خواهیم با استفاده از قضیه Yao Min Max نشان دهیم این مسئله در زمان محبّر از $n^{0.694}$ حل نمی شود:

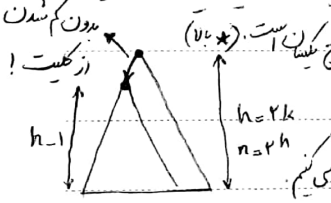
تعریف می کنیم $n_i = \begin{cases} p = \frac{3-\sqrt{5}}{2} & 1 \\ 1-p & 0 \end{cases}$ توزیع X روی ورودی مسئله را به صورت ،، توجه کنید که دو درخت روبرو یکسانند:



★ خاصیت این توزیع با درخت NOR این است که توزیع احتمال روی سطح i بالاتر درخت هم همین خواهد بود؛ چون: $(1-p)^2 = p$

می خواهیم $\min_a \mathbb{E}[c(a, X)]$ را برای الگوریتم های قطعی a محاسبه کنیم.

برای هر الگوریتم قطعی برای این مسئله می توان فرض کرد که الگوریتم ابتدا بررسی های مربوط به یک شاخه را به طور کامل انجام می دهد و بعد به سراغ بررسی آن شاخه می رود. (ممکن است الگوریتم اینطور نباشد و بررسی ها را قرعه وار انجام دهد اما می توان بررسی ها را به این شکل مرتب کرد. اثبات دقیق این موضوع کمی سخت است!)



بنابراین اگر $w(h)$ همان $\mathbb{E}(c(a, X))$ برای درختی با ارتفاع h باشد داریم:

$$w(h) = w(h-1) + (1-p)\,w(h-1) = (2-p)\,w(h-1) = (2-p)^h \underbrace{w(0)}_{1} \simeq (2-p)^{\log_2 n} = n^{0.694}$$

Subject:
Date

treap: درختی که در آن دوتایی های (x_i, y_i) به این صورت قرار گرفته اند که x_i ها BST ساخته اند و y_i ها heap ساخته اند.

توجه کنید که اگر مقادیر x_i ها و مقادیر y_i ها متمایز باشند، treap این مقادیر به صورت یکتا مشخص می شود (در واقع با شروع از ریشه می توان treap را ساخت.)

می خواهیم با استفاده از ایده تصادفی سازی و ایده treap، یک BST برای مقادیر $x_1, \dots, x_n$ بسازیم (به این صورت که مقادیر تصادفی y_i که متمایز باشند را به x_i ها متناظر کنیم) و treap دوتایی های به دست آمده را بسازیم. ادعا می کنیم اگر y_i ها توزیع یکنواخت بین صفر و یک داشته باشند، امید ریاضی ارتفاع treap، $\lg n$ است.

فرض کنید: $x_1 < x_2 < \dots < x_n$. اگر $depth(x_i)$ ارتفاع x_i در درخت باشد، می خواهیم $\mathbb{E}(\max(depth(x_i)))$ برابر با $O(\log n)$ باشد.

ابتدا سعی می کنیم $\mathbb{E}(depth(x_i))$ را به دست آوریم.

$$X^i_j = \begin{cases} 1 & \text{اگر } x_j \text{ جد } x_i \text{ باشد} \\ 0 & \text{اگر } x_j \text{ جد } x_i \text{ نباشد} \end{cases} \Rightarrow depth(x_i) = \sum_{j=1}^{n} X^i_j$$

$$\mathbb{E}(depth(x_i)) = \sum_{j=1}^{n} \mathbb{E}(X^i_j) = \sum_{j=1}^{n} Pr(X^i_j = 1)$$

★ X^i_j برابر با یک است اگر و فقط اگر بین مقادیر $\{y_i, \dots, y_{j+1}, y_j\}$، y_j ماکزیمم باشد. (چرا؟ به طریقه یکتای ساخته شدن treap توجه کنید)

$$\Rightarrow Pr(X^i_j) = \frac{1}{|i-j|+1} \Rightarrow \mathbb{E}(depth(x_i)) = \sum_{j=1}^{i-1} \frac{1}{i-j+1} + \sum_{j=i+1}^{n} \frac{1}{j-i+1} = H_i + H_{n-i+1}$$

حال برای بررسی $\mathbb{E}(\max(depth(x_i)))$ اینطور عمل می کنیم: اگر نشان دهیم $Pr(depth(x_i) \geq V \log n) \leq \frac{1}{n^k}$، خواهیم داشت: (این مقدار را می توان در ابتدا متغیر قرار داد و در ادامه مقدار مطلوب را یافت.)

$$Pr(depth(x_1) \geq V \log n \vee \dots \vee depth(x_n) \geq V \log n) \leq n \times \frac{1}{n^k} = \frac{1}{n^{k-1}}$$

و در ادامه مثل دفعات قبل $\mathbb{E}(\max(\dots))$ عمل می کنیم.

برای بررسی $Pr(depth(x_i) \geq V \log n) \leq \frac{1}{n^k}$ از نامساوی چرنوف استفاده می کنیم. توجه کنید که X^i_j مستقل از $X^i_{j'}$ است (چرا؟ بدیهی نیست، باید بررسی کرد!)

$$Pr(X > (1+\delta)\mu) \leq \left(\frac{e^\delta}{(\delta+1)^{\delta+1}}\right)^{\mu} \leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^{H_n} \dots$$

$H_i + H_{n-i+1} \geq H_n$

Subject:
Date

حالا هم با مقدار جدید x وارد شد، مقدار تصادفی y را به آن متناظر می‌کنیم و آن را وارد streap می‌کنیم. insert کردن در streap به این صورت است، بنا بر اساس y_i آن را تا برگ می‌بریم سپس بدون این که خاصیت BST از بین برود آن را بالا می‌بریم تا مقدار y آن در جایگاه درست قرار گیرد.

(برای این کار نسبت درخت سیاه-قرمز عمل می‌کنیم. البته خیلی ساده‌تر از درخت سیاه-قرمز است!)

توجه کنید که در عمل نمی‌توان y_i ها را به طور تصادفی بین مقادیر حقیقی متمایز انتخاب کرد، بلکه در عمل y_i عددی از بین m مقدار مختلف انتخاب خواهد شد. در این صورت $\Pr(X^i_j = 1)$ دقیقاً برابر با $\frac{1}{|i-j|+1}$ نیست. (زیرا ممکن است مقادیر y با هم برابر باشند، این حالت در قضیه $\frac{1}{|i-j|+1}$ لحاظ نشده است.)

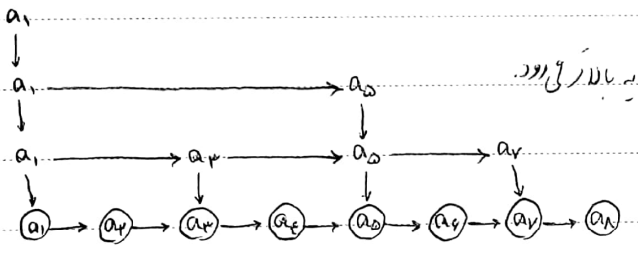
اگر y_t ها را از بین مقادیر $\{1, 2, \dots, m\}$ انتخاب کنیم، احتمال این که y ها کاملاً متمایز باشند از بین مقادیر $y_1, \dots, y_t$ برابر است با:

$$\sum_{k=2}^{m} \left(\frac{1}{m}\right)\left(\frac{k-1}{m}\right)^{t-1} = \frac{1}{m^t}\sum_{k=2}^{m}(k-1)^{t-1} \simeq \frac{1}{m^t}\cdot\frac{(m-1)^t}{t} = \frac{\left(1-\frac{1}{m}\right)^t}{t}$$

(y_k برابر با k باشد، بقیه کوچکتر از k باشند)

اگر m را (برحسب t) مقدار بزرگی در نظر بگیریم، می‌توان حاصل را $\frac{1}{t}$ در نظر گرفت.

شبه skip-list: شبه BST با این تفاوت که از پایین به بالا ساخته می‌شود، فقط در برگ‌ها رکورد وجود دارد و مانند BST، جستجو را ساده می‌کند.



skip-list را به صورت تصادفی به این صورت می‌سازیم که هر برگ با احتمال $\frac{1}{2}$ به لایه بالاتر می‌رود.

در این صورت کران بالا برای ارتفاع درخت را این‌طور تخمین می‌کنیم:

$$\Pr(h(a_i) \geq s) = \frac{1}{2^s} \qquad \text{ارتفاع: } h = \max_{i=1:n}(h(a_i))$$

$$\Pr(h \geqslant c\lg n) = \Pr(h(a_1) \geqslant c\lg n \vee \dots \vee h(a_n) \geqslant c\lg n) \leqslant \sum_{i=1}^{n}\Pr(h(a_i) \geqslant c\lg n) = \sum \frac{1}{2^{c\lg n}} = \frac{n}{n^c} = \frac{1}{n^{c-1}}$$

$$\mathbb{E}(h) = \sum_{i=1}^{\infty} i\Pr(h=i) = \underbrace{\sum_{i=1}^{2\lg n} i\Pr(h=i)}_{O(\lg n)} + \sum_{i=2\lg n}^{n} i\Pr(h=i) + \underbrace{\sum_{i=n}^{2n} i\underbrace{\Pr(h=i)}_{<\frac{n}{2^{2n}}}}_{\frac{2n^3}{2^{2n}}} + \underbrace{\sum_{i=2n}^{3n} i\Pr(h=i)}_{n\times 3n\times\frac{n}{2^{2n}}} + \dots$$

Subject:
Date

همچنین در هر سطح، امید ریاضی تعداد حرکت ها رو به جلو، ۲ است (چون دو حالت با احتمال $\frac{1}{2}$ است.) بنابراین سه برابر کردن مجموعاً $O(\lg n)$ است.

بنابراین درج و حذف و سرچ در $O(\lg n)$ قابل انجام است.

coppon collector: حداقل چند توپ داخل ظرف بریزیم تا حداقل به احتمال $1-\frac{1}{m}$ داخل هر جعبه توپ وجود داشته باشد؟ (تعداد جعبه ها m است و هر توپ با احتمال یکسان داخل جعبه ای مختلف می رود.)

$$\frac{1}{m} \geq Pr(\text{ظرف ۱ = ۰ یا ظرف ۲ = ۰ یا ... یا ظرف m = ۰}) \geq \sum Pr(\text{ظرف i = ۰}) = m\left(1-\frac{1}{m}\right)^n$$

$$n \geq 2m \ln m \iff m\left(1-\frac{1}{m}\right)^n \leq m\left(\frac{1}{e}\right)^{2\ln m} = \frac{1}{m}$$

راه دوم: $\mathbb{E}(\text{تعداد توپ لازم برای پر شدن ظرفی}) = \sum_{i=1}^{m} X_i$

X_i: تعداد دفعات توپ انداختن برای پر شدن ظرف $i+1$ ام بعد از پر شدن i ظرف $\Rightarrow$ X_i دوجمله‌ای با احتمال $\frac{m-i}{m}$ $\Rightarrow \mathbb{E}[X_i] = \frac{m}{m-i}$

$$= \sum \frac{m}{m-i}$$

Subject:
Date

Random Walk: قدم زدن تصادفی:

H_{uv} = expected time to reach v from u

C_u = expected cover time (time to reach all vertices and returning) by starting from u

$C(G) = \max C_u$

در جلسه قبل با استفاده از یک لم که آن را اثبات نکردیم، نشان دادیم: $C(G) \le 2m(n-1)$ (مشروح جلسه قبل در اسلاید)

در این جلسه لم اثبات نشده را اثبات می‌کنیم

لم: اگر در گراف به جای هر یال، یک مقاومت یک اهمی قرار دهیم، آنگاه داریم: $H_{uv} + H_{vu} = 2mR_{uv}$

اثبات: ابتدا برای H_{uv} یک رابطه بازگشتی می‌نویسیم: (اگر به راحتی نگریم، این رابطه یک $n-1$ معادله $n-1$ مجهول می‌دهد)

$$H_{uv} = 1 + \frac{1}{d(u)} \sum_{ux \in G} H_{xv}$$

حالتی را فرض کنید که در مدار، به هر رأس به اندازه درجه‌اش جریان تزریق می‌کنیم و از رأس v کل جریان $2m$ را خارج می‌کنیم. (توجه کنید که به خود v نیز به اندازه درجه‌اش جریان تزریق کرده‌ایم.) اگر ولتاژ بین u و v را با Q_{uv} نشان دهیم و جریان را با I_{uv}، داریم:

$$d(u) = \sum_{ux \in G} I_{ux} = \sum_{ux \in G} Q_{ux} / R_{ux=1} = \sum_{ux \in G} (Q_{uv} - Q_{xv}) \Rightarrow Q_{uv} = \frac{1}{d(u)} \left(d(u) + \sum_{ux \in G} Q_{xv} \right) = 1 + \frac{1}{d(u)} \sum_{ux \in G} Q_{xv}$$

این رابطه هم یک $n-1$ معادله $n-1$ مجهول برای Q_{xv} ها می‌سازد (با فرض ثابت گرفتن v) که دقیقاً مثل H_{uv} است. بنابراین: $H_{uv} = Q_{uv}$

همین طور برای محاسبه H_{vu}، جریان را از رأس u خارج می‌کنیم. (البته باید به جریان‌ها را منفی کنیم.) حال با استفاده از جمع آثار به مداری می‌رسیم که جریان $2m$ از رأس u وارد و از رأس v خارج می‌شود. در این مدار، ولتاژ بین u و v، $2mR_{uv}$ است. ■

Subject:
Date

در گراف کامل:

$$H_{uv} = \frac{1}{n-1} \times 1 + \frac{n-2}{n-1}(1 + H_{uv}) \Rightarrow H_{uv} = n-1 \qquad C(K_n) = O(n \lg n)$$

↓ مشدد قوی و ضعیف

فرض کنید در گرافی در حال قدم زدن تصادفی هستیم و در هر واحد زمان، یک یال را طی می‌کنیم. $P_u(t)$ احتمال بودن در رأس u در زمان t است.

مثال: شروع از رأس ۱

time =	۱	۲	۳	۴
①	۱	۰	$\frac{1}{2}$	...
②	۰	$\frac{1}{2}$	$\frac{1}{4}$	...
③	۰	$\frac{1}{2}$	$\frac{1}{4}$	...
	$P_{(1)}$	$P_{(2)}$	$P_{(3)}$	

$$P(t+1) = M \cdot P(t)$$

↓ ماتریس احتمال حرکت بین رئوس

اگر گراف قویاً همبند باشد و ب.م.م اندازه دور‌ها یک باشد (دور نامقدار) $\lim_{t\to\infty} P_u(t)$ همگرا است. در این صورت: $\lim_{t\to\infty} P(t) = \pi \Rightarrow \pi = M.\pi$

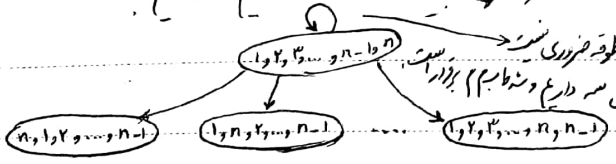
که یک جواب معادله $\pi = M.\pi$، واضحاً مقدار روی‌رو است. و چون تنها یک جواب دارد، تنها جواب ممین است:

$$\pi = \left[\frac{d(v_1)}{2m}, \frac{d(v_2)}{2m}, \ldots, \frac{d(v_n)}{2m}\right]$$

توجه کنید که در گراف بدون جهت برای یک بودن ب.م.م دور‌ها، کافیست دور فرد داشته باشیم.

همچنین توجه کنید که اگر درجه رئوس برابر باشد، داریم: $\pi = \left[\frac{1}{n}, \ldots, \frac{1}{n}\right]$ این نکته برای sampling بسیار بسیار مفید است. فرض کنید می‌خواهیم اندیس یک زیردرخت دلخواه پوشای یک گراف، یکی را به تصادف انتخاب کنیم. یک راه که به شدت این کار را ساده می‌کند، استفاده از قدم زدن تصادفی است، به این صورت که هر زیردرخت پوشا را معادل با یک رأس در نظر می‌گیریم و شرطی برای مجاور بودن دو رأس (دو زیردرخت پوشا) تعیین می‌کنیم که درجه همه رئوس برابر باشد، گراف همبند شود و دور فرد داشته باشد.

مثلاً برای ساختن جایگشت تصادفی به این روش کافیست به روش مثل روش زیر بین حالات حرکت کنیم. می‌توانیم به سادگی کار را از جایگشت



ترتیب شروع کنیم:

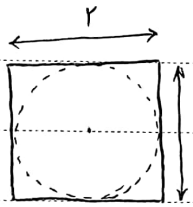
Subject:
Date

برای مثال جایگشت امید ریاضی زمان رسیدن به توزیع یکنواخت بین جایگشت ها را بررسی می‌کنیم (تا به حال صرفاً ثابت کردیم که در زمان بی‌نهایت به توزیع یکنواخت می‌رسیم.) فرض کنید یک ستاره در ابتدای جایگشت داریم، هر عددی که قرار بود به کنار ستاره منتقل شود را به سمت چپ ستاره می‌بریم.

امید ریاضی زمان آمدن اولین عدد به سمت چپ ستاره، n است، دومین عدد $\frac{n}{2}$، سومی $\frac{n}{3}$ و ... بعد از آمدن همه اعداد به سمت چپ ستاره، جایگاه آن‌ها کاملاً تصادفی است و امید ریاضی زمان این (تقریباً) $O(n\log n)$ است. $n+\frac{n}{2}+\frac{n}{3}+\dots+\frac{n}{n}$

گرفتی چی شد؟!!! تعداد جایگشت‌ها $n!$ تاست.
$n!$ بار قدم زدن تصادفی که خیلی کم از تعداد رئوس (تعداد جایگشت‌هاست) $n\log n$ بار به توزیع یکنواخت روی رئوس (جایگشت‌ها) رسیدیم!

تخمین عدد π با استفاده از یک روش تصادفی:



در داخل دایره دورو یک نقطه به صورت تصادفی انتخاب می‌کنیم اگر فاصله آن از مرکز کمتر از یک بود Z_i را برابر با یک و گرنه برابر با صفر قرار می‌دهیم. این آزمایش را n بار تکرار می‌کنیم و مقادیر Z_i را می‌سازیم. امید ریاضی Z_i، $\frac{\pi}{4}$ است. در پایان مقدار $\frac{4}{k}\sum_{i=1}^{k} Z_{ki}$ را به عنوان مقدار تخمینی π گزارش می‌کنیم. k چقدر باشد که شرط دورو برقرار باشد؟ $Pr\left(\left|\frac{4}{k}\sum_{i=1}^{k} z_i - \pi\right| \geq \varepsilon\pi\right) \leq \delta$

$\Leftarrow$ کافیست ... $\Leftrightarrow Z=\sum_{i=1}^{k} z_i$ ، $Pr\left(\left|Z-\frac{k\pi}{4}\right| \geq \varepsilon\frac{k\pi}{4}\right) \leq \delta$

با استفاده از نامساوی چرنوف به دست می‌آوریم که شرط $k \geq \frac{12\ln\left(\frac{1}{\delta}\right)}{\pi\varepsilon^2}$ کافیست. (در این رابطه می‌توان به جای π یک کران پایین مثل ۳ قرار دهیم.)

قضیه: به طور کلی طبق نامساوی چرنوف برای متغیرهای تصادفی $i.i.d$ (برنولی مستقل و هم توزیع) $X_1, X_2, \dots, X_m$ که $E(X_i)=\mu$ داریم:

$m \geq \frac{3\ln\left(\frac{1}{\delta}\right)}{\mu\varepsilon^2} \Rightarrow Pr\left(\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \varepsilon\mu\right) \leq \delta$

Subject:
Date

DNF counting:

یک عبارت منطقی به صورت SOP داریم (که طول هر پرانتز ممکن است هر چقدر باشد و محدودیتی ندارد) می‌خواهیم تعداد تخصیص‌هایی که این عبارت را صحیح می‌کند بیابیم:

$Q = (\quad \wedge \quad \wedge \quad) \vee (\quad \wedge \quad \wedge \quad \wedge \quad \wedge \quad) \vee (\quad \wedge \quad) \vee \dots$ (طبیعتاً not هم می‌تواند باشد)

این مسئله NP-hard است؛ چون 3SAT به این مسئله کاهش می‌یابد. در مسئله 3SAT کل عبارت را نقیض می‌گیریم که یک عبارت به این صورت که به شکل بالا می‌شود، اگر جواب DNF counting 2^n بود یعنی همواره درست است و بنابراین عبارت اولیه همواره غلط (unsatisfiable) بوده.

می‌خواهیم یک الگوریتم تصادفی با (ε,δ)-approximation برای مسئله DNF counting ارائه دهیم.

```
X ← 0
for k=1 to m
    generate a random assignment for n variables
    if the assignment satisfies Q: X ← X+1
return (n/m) × 2^n
```

طبق قضیه صفحه قبل، برای (ε,δ)-approximation باید الگوریتم را با $m \geq \frac{3\ln(\frac{2}{\delta})}{\varepsilon^2 \mu}$ بار اجرا کنیم. μ را توجه کنید که: $\mu = \frac{\text{تعداد ass. مقبول}}{2^n}$

بنابراین اگر تعداد ass. قابل قبول کم باشند، $m = O(2^n)$ خواهد بود. برای حل این مشکل به صورت زیر عمل می‌کنیم:

فرض کنید $F = \overbrace{(\quad \wedge \quad)}^{C_1} \vee \overbrace{(\quad \wedge \quad)}^{C_2} \vee \dots \vee \overbrace{(\quad \wedge \quad)}^{C_k}$ تعریف می‌کنیم:

$H = \{(i,a) \mid a \text{ satisfies } C_i\}$ (i: یک assignment)

$SC_i = \{(i,a) \mid a \text{ satisfies } C_i\}$ (این بین است!)

$H = \bigcup SC_i \qquad |H| = \sum |SC_i|$

$S = \{(i,a) \mid a \text{ satisfies } C_i, \text{ for any } j<i;\ j \text{ does not satisfies } C_j\}$ ← جواب DNF Count $\qquad |H| \leq k|S|$

Subject:
Date

برای تخمین کردن مجاسبه اندازه SC_i ! ساده (و در زمان چندجمله‌ای) است. تخمین با نمونه یک نمونه تصادفی در SC_i نیز ساده است.

(چون تعدادی از n_i ! ثابت و بقیه متغیر و بقیه n_i ! آزادند.) اندازه H نیز حاصل جمع اندازه SC_i ! است. می‌خواهیم نسبت $\frac{|S|}{|H|}$ را بیابیم.

ابتدا توجه کنید که این نسبت حداکثر k^k است. از آنجا که طول ورودی وابسته به k است، طبیعتاً اندازه k چندجمله‌ای است. (اصلاً چندجمله‌ای

پس قضیه (ε,δ)approx درآنجا معقول است.

بودن جواب بعضی رحسب n^k، چندجمله‌ای باشد! پس خود k قطعاً چندجمله‌ای فرض می‌شود!!) حال نسبت $\frac{|S|}{|H|}$ را با نمونه‌گیری می‌یابیم.

ابتدا با نسبت وزن‌دار (براساس اندازه) یکی از SC_i ! را انتخاب می‌کنیم و سپس درون SC_i یکی از اعضا را به تصادف انتخاب می‌کنیم (و گفتیم که

این کار ساده‌ای است.) سپس به سادگی بررسی می‌کنیم که آن عضو در S هست یا نه. سپس با یافتن آن نسبت، مقدار خود |S| را تخمین می‌زنیم.

Independent Set:

می‌خواهیم از بین همه زیرگراف‌های القایی G، تعداد آنهایی را بیابیم که هیچ یالی ندارند (یعنی در واقع روش انتخاب نشده برای زیرگراف Indep. Set بوده‌اند.) $\Omega(G)$

روش نمونه‌گیری مثل مسئله قبل این مشکل را دارد که ممکن است طبق قضیه (ε,δ) approx.، تعداد نمونه‌گیری خالی برای رسیدن به تقریب

خوبی از جواب لازم باشد. پس باید یک ایده بزنیم! فرض کنید گراف اولیه G_m باشد و با حذف یک یال از G_i، G_{i-1} ساخته می‌شود. (G_0 هم گراف بی‌یال است.)

$$\Omega(G)=\frac{\Omega(G_m)}{\Omega(G_{m-1})}\times\frac{\Omega(G_{m-1})}{\Omega(G_{m-2})}\times\dots\times\frac{\Omega(G_1)}{\Omega(G_0)}\times\underbrace{\Omega(G_0)}_{2^n}=r_m\times r_{m-1}\times\dots\times r_2\times r_1\times 2^n \qquad r_i=\frac{\Omega(G_i)}{\Omega(G_{i-1})}$$

سعی می‌کنیم r_i را تخمین بزنیم. طبق اثبات کتاب اگر بتوانیم r_i ! را با $(\frac{\varepsilon}{2m},\frac{\delta}{2m})$approx تخمین بزنیم، نتیجه حاصل $\Omega(G)$ با تقریب (ε,δ)approx خواهد بود.

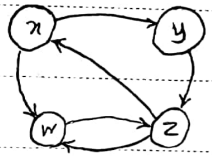
تخمین r_i را به این صورت انجام می‌دهیم. از independent set های G_{i-1} (که تعدادشان $\Omega(G_i)$ است) نمونه می‌گیریم و بررسی می‌کنیم که آیا در G_i هم indep.set هستند یا نه.

توجه کنید که r_i حداکثر ۲ است.

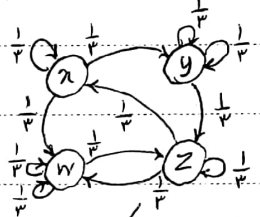
Subject:
Date

تنها قسمت سخت، نمونه‌گیری (به صورت یکنواخت) از بین همه independent set های G_{i-1} است. این کار را با random walk انجام می‌دهیم.

گراف زیر را داریم. می‌خواهیم احتمال حرکت روی یال‌های آن را طوری تعریف کنیم که stationary distrib. آن $\pi = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ شود:



همانطور که قبلاً دیدیم اگر درجه رئوس برابر باشد، رابطه $P\pi = \pi$ طبق یکنواخت است. پس اگر طوقه‌هایی به صورت زیر



بگذاریم، به π یکنواخت می‌رسیم: (البته توضیح قبلی مربوط به گراف‌های بدون جهت بود. الآن هم منظورمان بدون جهت بود.)

بنابراین $P(x,y)$ را به صورت روبرو تعریف می‌کنیم:

$$P(x,y) = \begin{cases} \frac{1}{n} & x \neq y, (x,y) \in E \\ 0 & x \neq y, (x,y) \notin E \\ 1 - \frac{\deg(x)}{n} & x = y \end{cases}$$

به طور کلی برای این که π یکنواخت باشد، طبق رابطه $P\pi = \pi$ کافیست داشته باشیم: $\pi_i P(i,j) = \pi_j P(j,i)$ (این شرط لازم نیست ولی شرط کافی (احتمالاً) خیلی به درد بخوری است.)

حالا به مسئله قبل برمی‌گردیم. می‌خواهیم با random walk از بین همه independent set های یک گراف، به صورت یکنواخت، نمونه‌گیری کنیم.

طبق ایده بالا کافیست گرافی بسازیم از node هایی که نماینده independent set ها هستند بسازیم. سپس با تعریف طوقه‌ها به صورت مناسب، کاری می‌کنیم π یکنواخت شود. نهایتا الگوریتم روبرو جواب می‌دهد: (این الگوریتم طوقه‌ها را در خودش به طور مناسب حساب می‌کند.)

X_0 is an arbitrary IS

To compute X_{i+1} from X_i

choose a vertex v uniformly random (توجه کنید که اگر مثلاً بخواهیم به node v را به هم وصل کنیم، در واقع در هر مرحله لازم است)

if $v \in X_i$: $X_{i+1} = X_i - \{v\}$

if $v \in X_i$:

if $X_i \cup \{v\}$ is an independent set: $X_{i+1} = X_i \cup \{v\}$ else $X_{i+1} = X_i$

کل independent set ها را محاسبه کنیم که این در زمان چندجمله‌ای ممکن نیست.

$\pi_i P(i,j) = \pi_j P(j,i)$

$\frac{1}{k} \quad \frac{1}{n} \quad \frac{1}{k} \quad \frac{1}{n}$

$k \leftarrow$ تعداد independent set ها است، n تعداد رئوس گراف اصلی است.

Subject
Date

یک مثال جالب دیگر [redacted] زیردرخت‌های پوشای یک گراف است که می‌توان با قدم زدن تصادفی از آن نمونه‌گیری تصادفی کرد. و ثابت شده که می‌توان

این قدم زدن را در زمان چندجمله‌ای انجام داد (مقاله این اثبات را می‌توان به عنوان ارائه کلاس آماده کرد.)

* برای π غیریکنواخت نیز اگر گراف بدون جهت باشد، کافیست وزن یال i و طوری

$$P(x,y)=\begin{cases}\frac{1}{M}\min\left(1,\frac{M_y}{M_x}\right) & x\neq y,\ (x,y)\in E\\ 0 & x\neq y,\ (x,y)\notin E\\ 1-\text{بقیه} & x=y\end{cases}$$

را به صورت روبرو مقدار بدهیم تا به M مورد نظر برسیم.

Subject

Date

Subject
Date

سه ماتریس $n \times n$ به نام‌های A، B و C را داریم. بررسی درستی $A \times B = C$ در حالت عادی به زمان $O(n^3)$ نیاز دارد. می‌خواهیم درستی این رابطه را در زمان $O(n^2)$ با احتمال خوبی تشخیص دهیم. الگوریتم روبرو را ارائه می‌دهیم:

select a random vector $r_{n \times 1}$ ← $(1, 0, \dots, 0)^T$

compare $(A \times B)r$ and Cr ← تولید کنید که با به این دو مقدار در $O(n^2)$ ممکن است

if $(A \times B) = Cr$ then return "$A \times B = C$"

else return "$A \times B \neq C$"

الگوریتم وقتی جواب No می‌دهد جواب درست است. بنابراین فقط وقتی جواب Yes می‌دهد احتمال خطا وجود دارد. می‌خواهیم نشان دهیم: $\Pr(ABr = Cr \mid AB \neq C) \leq \frac{1}{2}$

$D = AB - C \neq 0 \Rightarrow \exists d_{ij} \neq 0 \;,\; Dr = 0 \Rightarrow \sum_{k=1}^{n} d_{ik} r_k = 0 \Rightarrow r_j = \frac{\sum_{k \neq j} d_{ik} r_k}{d_{ij}}$ ← بیان درایه‌ای که غیرصفر بود

احتمال این که این تساوی برقرار باشد حداکثر $\frac{1}{2}$ است؛ زیرا r_j مستقل از D (که از روی A و B و C به دست می‌آید) و بقیه مقادیر بردار r، یکی از دو مقدار ۰ یا ۱ است (با احتمال $\frac{1}{2}$) ولی مقدار سمت راست حتماً یکی از مقادیر ۰ یا ۲ یا حتی مقادیر دیگر را اتخاذ کرده است. ■

همچنین اگر به جای ۲، k مقدار متمایز بتواند بگیرد، احتمال خطا به جای $\frac{1}{2}$، می‌شد $\frac{1}{k}$.

Finger Print: بسیاری از مواقع می‌خواهیم تساوی دو چیز را بررسی کنیم که بررسی دقیق آن‌ها سخت است. در این مواقع تابعی که اصطلاحاً به آن $\text{Fing}(x)$ می‌گوییم روی دو قسمت اعمال می‌کنیم که اگر جواب هر دو برابر بود حدس می‌زنیم که مقادیر اولیه نیز برابر بوده‌اند.

با تکرار عملیات بالا احتمال خطا کم و کم‌تر می‌شود.

Subject
Date

فرض کنید می‌خواهیم بررسی کنیم که دو چندجمله‌ای $Q_1(x)$ و $Q_2(x)$ برابر هستند یا نه. به این صورت عمل می‌کنیم که عددی را در $Q_1(x)$ و $Q_2(x)$ قرار می‌دهیم و بررسی می‌کنیم که آیا مقدار به دست آمده برابر است یا نه. جواب وقتی اشتباه است که Q_1 و Q_2 برابر نباشند اما نقطه بررسی شده، $Q_1(x) - Q_2(x)$ باشد. برای این که این احتمال را کنترل کنیم عدد تصادفی را از مجموعه‌ای با اندازه دو برابر تعداد ریشه‌های $Q_1(x)$ و $Q_2(x)$ (ماکزیمم درجه) انتخاب می‌کنیم، در این صورت شرط $\Pr(\text{Fing}(Q_1)=\text{Fing}(Q_2) \mid Q_1=Q_2) \geq \frac{1}{2}$ که شرط خوبست (۱) برقرار خواهد بود.

اما چندجمله‌ای‌های چندمتغیره چطور؟ در آنجا تعداد ریشه‌ها بی‌نهایت است! انسان می‌دهیم که اگر مقادیر $r_1, r_2, \dots, r_n$ را به طور یکنواخت و مستقل از مجموعه $|S|$ انتخاب کنیم، خواهیم داشت: (d ماکزیمم درجه Q_1 و Q_2)

$$\Pr\left(Q_1(r_1,r_2,\dots,r_n) = Q_2(r_1,r_2,\dots,r_n) \mid Q_1 \neq Q_2\right) \leq \frac{d}{|S|}$$

این حکم را با استقرا روی n ثابت می‌کنیم. فرض کنید $Q = Q_1 - Q_2$ و درجه x_n در Q برابر با k باشد. اگر Q را بر اساس درجه x_n مرتب کنیم داریم:

$$Q(x_1,x_2,\dots,x_n) = \sum_{i=0}^{k} H_i(x_1,x_2,\dots,x_{n-1}).x_n^i$$

دو حالت را باید بررسی کنیم. حالتی که $H_k(r_1,r_2,\dots,r_{n-1})$ برابر با صفر است و حالتی که برابر با صفر نیست. ($\bar{E}$) (E) در حالتی که برابر با صفر نیست، $Q(r_1,r_2,\dots,r_{n-1},x_n)$ یک چندجمله‌ای از درجه یک است که همان طور که در بالا گفته شد عدد رندم x_n از مجموعه S، به احتمال $\frac{k}{|S|}$ حداکثر $\frac{d_{H_k}}{|S|}$ است. (توجه کنید که $d - k \geq d_{H_k}$) دچار خطا می‌شود. بنابراین احتمال رخ دادن حالت دیگر نیز طبق فرض استقرا حداکثر است. بنابراین:

$$\Pr(\text{خطا}) = \underbrace{\Pr(\text{خطا} \mid E)}_{\leq 1}\underbrace{\Pr(E)}_{\leq \frac{d_{H_k}}{|S|}} + \underbrace{\Pr(\text{خطا} \mid \bar{E})}_{\leq \frac{k}{|S|}}\underbrace{\Pr(\bar{E})}_{\leq 1} \leq \frac{d_{H_k}}{|S|} + \frac{k}{|S|} \leq \frac{d}{|S|} \blacksquare$$

Subject
Date

فرض کنید Alice و Bob هر کدام یک عدد n بیتی دارند و می‌خواهند با ارسال پیام تا حد ممکن کوتاهی به یکدیگر، چک کنند که آیا عددشان با هم برابر است یا نه. می‌توان با استدلال ساده کوتری نشان داد که برای اطمینان ۱۰۰٪ لازم است Alice کل n بیت را ارسال کند.

توجه کنید که در روش Finger Print احتمال خطا در رد کردن نمی‌تواند صفر باشد. یعنی وقتی الگوریتم می‌گوید نابرابری نه!

اما اگر بخواهیم مثل قبیه مسائل Finger Print احتمال خطا در تایید را به کمتر از $\frac{1}{2}$ برسانیم به طریق زیر عمل می‌کنیم:

Alice با تعیین عدد خود را به یک عدد اول p (که $O(\log n)$ بیت دارد) برای Bob می‌فرستد و باقی عدد خود را با آن چک می‌کند. در این حالت خطا زمانی اتفاق می‌افتد که $x-y$ به p بخش‌پذیر باشند. توجه کنید که $x-y$ حداکثر n بیتی است و حداکثر n عامل اول دارد.

بنابراین اگر Alice عدد اول خود را از یک مجموعه $2n$ تایی از اعداد اول انتخاب کند، احتمال این که یکی از عوامل اول $x-y$ را انتخاب کرده باشد، کمتر از $\frac{1}{2}$ است. پس به کران خود قطر رسیدیم. اما آیا n تا عدد اول داریم که تعدادشان $O(\log n)$ باشد؟ طبق قضیه‌ای، تعداد اعداد اول در بازه $\{1,\dots,N\}$، $\frac{N}{\ln N}$ تاست. بنابراین اگر بگیریم: $N = n \cdot \ln^2 n$، به تعداد n تا عدد اول در بازه $\{1,\dots,N\}$ خواهیم داشت

و توجه کنید که N یک عدد $O(\log n)$ بیتی است.

Subject

Date

Subject

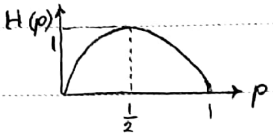
Date

Entropy: آنتروپی:

$H(X) = \mathbb{E}\left(\lg_2 \frac{1}{Pr(X=x)}\right) = -\sum_x Pr(X=x).\lg_2(Pr(X=x))$ ← آنتروپی به مقدار عددی متغیر تصادفی بستگی ندارد.

$X = \begin{cases} 1 & p \\ 0 & 1-p \end{cases}$ شیر / خط

$H(X) = -(p.\lg_2 p + (1-p)\lg_2(1-p))$



یا $H(p)$ نزدیک ۱ می‌شود

می‌خواهیم نشان دهیم با انداختن سکه با احتمال p، می‌توان $H(p)$ تا بیت رندم تولید کرد!!! بیت رندم یعنی بیتی که احتمال ۰ یا ۱ بودن آن دقیقاً $\frac{1}{2}$ است.

اگر X_1 و X_2 مستقل باشند: $H(X_1, X_2) = H(X_1) + H(X_2)$

مثال: آنتروپی تاس، $\lg_2 6$ است. از طرف دیگر اگر تاسی را n بار بیندازیم، با خروجی آن می‌توانیم $n.\lg_2 6$ بیت رندم تولید کنیم.

می‌خواهیم با داشتن یک مقدار تصادفی، بیت تصادفی تولید کنیم (هرچه بیشتر بهتر) به این کار Entract کردن می‌گوییم $(Ext(.))$

فرض کنید عددی تصادفی با توزیع نامعلوم در بازه $\{0,1,\dots,m-1\}$ داریم. مثلاً برای $m=10$ به صورت زیر عمل می‌کنیم:

یک عدد سه بیتی تولید می‌کنیم ← $0,1,\dots,7$ — یک عدد یک بیتی تولید می‌کنیم ← $8,9$

در حالت کلی در بازه $\{0,\dots,m-1\}$ اگر خروجی در بازه $0,\dots,2^\alpha-1$ بود (بزرگترین α) یک عدد α بیتی تولید می‌کنیم. در غیر این صورت بازه باقیمانده را به همین صورت جلو می‌کنیم و تعداد بیت‌های رندم لازم را تولید می‌کنیم. می‌توان با استقرا نشان داد امید ریاضی تعداد بیت‌های رندم تولید شده بزرگتر از $\lfloor \log_2 m \rfloor - 1$ است.

حال فرض کنید سکه‌ای با احتمال p داریم و آن را n بار می‌اندازیم. می‌خواهیم با خروجی آن چند بیت رندم تولید کنیم. مثلاً برای $n=2$ به صورت زیر عمل می‌کنیم:

هیچ بیت رندمی تولید نمی‌کنیم! ← TT, HH — یک بیت رندم تولید می‌کنیم ← HT, TH

Subject
Date

در حالت کلی به این صورت عمل می‌کنیم. می‌دانیم به حالاتی که j بار سکه رو بیاید و $n-j$ بار پشت با احتمال $p^j(1-p)^{n-j}$ رخ می‌دهد. بنابراین این حالات را به تعداد $\binom{n}{j}$ از $\{0,1,\dots,\binom{n}{j}-1\}$ متناظر می‌کنیم و سپس مثل قبل از این تعداد بیت رندم تولید می‌کنیم. برای هر $n > j > 0$ همین کار را می‌کنیم.

امید ریاضی تعداد بیت‌های رندم تولید شده از این روش برابر است با: $\sum_{j=1}^{n-1} p^j(1-p)^{n-j}\binom{n}{j}\lg_2\binom{n}{j}$

قضیه: برای هر $\delta > 0$، می‌توان n بزرگی یافت که: $\sum_{j=1}^{n-1} p^j(1-p)^{n-j}\binom{n}{j}\lg_2\binom{n}{j} \geq (1-\delta)\, n\, H(p)$

توجه کنید که $nH(p)$ آنتروپی آزمایش ماست و این قضیه در واقع می‌گوید این الگوریتم تقریباً به اندازه آنتروپی می‌تواند بیت رندم تولید کند.

اثبات قضیه بالا را به طور کلی بررسی نمی‌کنیم اما برای اثبات از چند نامساوی احتمالاتی استفاده شده که مهم‌ترین آن این است: $\frac{2^{nH(p)}}{n+1} \leq \binom{n}{np} \leq 2^{nH(p)}$

قضیه: امید ریاضی تعداد بیت‌های رندم نمی‌تواند بیشتر از آنتروپی باشد.

اثبات: پیشامدی که با احتمال q رخ می‌دهد، حداکثر می‌تواند به یک عدد رندم $\log_2\frac{1}{q}$ بیتی متناظر شود؛ چون اگر [redacted] این پیشامد به یک عدد k بیتی متناظر شود، باید $2^k - 1$ پیشامد (یا مجموعه پیشامد) دیگر نیز وجود داشته باشند که به تعداد k بیتی متناظر شوند و می‌دانیم مجموع این احتمالات حداکثر ۱ است؛ پس: $2^k q \leq 1 \Rightarrow k \leq \lg_2\frac{1}{q}$. (با احتمال دقیقاً q)

حال امید ریاضی تعداد بیت‌های رندم [redacted] تولید شده برابر است با: $H(X) = \sum_{\delta} Pr(\delta)\lg_2\frac{1}{q} \geq \sum_{\text{همه پیشامدهای } \delta} Pr(\delta) \times (\text{تعداد بیت تولیدی پیشامد } \delta)$

بنابراین روش فوق، بهترین روش برای تولید بیت تصادفی از روی تعدادی متغیر تصادفی مستقل و هم‌توزیع است که به اندازه آنتروپی (که کران بالاست) بیت تصادفی تولید می‌کند.

Subject
Date

Hash:

می‌خواهیم بازه بزرگ مثل مجموعه $\{0,1,\dots,m-1\}$ را به مجموعه کوچکتری مثل $\{0,1,\dots,n-1\}$ نگاشت کنیم به طوری یکنواخت.

حالت ایده‌آل این است که یک تابع را به صورت تصادفی از بین همه n^m تابع ممکن انتخاب کنیم. اما چون بسیاری از توابع ممکن بین n^m تابع خوش فرم نیستند، این کار بسیار سخت است. بنابراین معمولاً توابع را از بین k تابع $\{h_1, h_2, \dots, h_k\}$ انتخاب می‌کنیم.

ویژگی یکنواخت بودن hash را به روش‌های متفاوتی می‌توان رلکس کرد؛ از جمله:

2-universal hash: $\forall x_1, x_2 : Pr\left(h(x_1) = h(x_2)\right) \le \frac{1}{n}$

2-strong universal hash: $\forall x_1, x_2, y_1, y_2 : Pr\left(h(x_1) = y_1, h(x_2) = y_2\right) = \frac{1}{n^2}$

نکته: یک 2-universal hash لزوماً 3-universal hash نیست.

در مسئله توپ و ظرف اگر توپ‌ها به طور کاملاً یکنواخت در ظروف توزیع شده باشند، ماکزیمم بود ظرف‌ها به احتمال حداقل $1-\frac{1}{n}$ کمتر از $\frac{\log n}{\log\log n}$ است.

اگر تابع hash کاملاً یکنواخت باشد نیست ماکزیمم بود در بُرد تابع باید به همین مقدار صورت باشد اما مثلاً در حالت 2-universal hash،

به خاطر رلکس شدن شرط یکنواختی ماکزیمم بود $\sqrt{2n}$ است (به احتمال حداقل $1-\frac{1}{n}$)

مثال: دسته‌ای از توابع hash را به این صورت می‌سازیم. عدد اول $p \ge m$ را در نظر می‌گیریم: $h_{a,b}(x) = (ax+b \bmod p) \bmod n$

$H = \{h_{a,b} \mid 1 \le a \le p-1, 0 \le b < p\} \quad |H| = p(p-1)$

Subject
Date

فرض کنید به صورت تصادفی یکی از اعضای H را انتخاب می‌کنیم. می‌خواهیم ببینیم تابع hash که به این صورت به دست می‌آید شرط 2-universal

را دارد یا نه:

$$\frac{\text{تعداد توابعی که: } h(x_1)=h(x_2)}{\text{کل توابع} = p(p-1)} \overset{?}{\geq} \frac{1}{n} \iff \text{تعداد توابعی که } h(x_1)=h(x_2) \overset{?}{\geq} \frac{p(p-1)}{n}$$

ابتدا توجه کنید که برای x_1 و x_2 ثابت و نیز $p > u, v \geq 0$، دقیقاً یک جفت (a,b) وجود دارد که:

$$ax_1 + b \bmod p = u$$
$$ax_2 + b \bmod p = v$$

(طبق یک قضیه نظریه اعدادی!)

در رابطه بالا، هر جایی که $v \equiv u \pmod n$ باشد، برای $h(x_1)=h(x_2)$ که تعداد دوتایی‌های (u,v) که در این شرایط صدق می‌کنند تقریباً برابر است با $p \cdot \frac{n}{p}$ (یا به طور دقیق‌تر عددی بین $n\lfloor \frac{n}{p} \rfloor$ و $n\lceil \frac{n}{p} \rceil$). بنابراین شرط 2-universal تقریباً (!) برقرار است. همچنین شرط 2-strong universal نیز برای این تابع hash وجود دارد که آن را در اینجا ثابت نمی‌کنیم.