Randomized Algorithm and Structure things

---

integer: $E(x) = \sum_{k=-\infty}^{+\infty} k \cdot P(X=k)$

$\downarrow$

Linearity of expectation: $E(aX+bY) = aE(X) + bE(Y)$

Markov's inequality: If $\begin{cases} X \geqslant 0 \\ a \geqslant 0 \end{cases}$ $P(X \geqslant a) \leq \frac{E(X)}{a}$

$\cancel{P(X \geqslant x) \leq E(x)}$

There are two kinds of randomized algorithms:

Las Vegas randomized algorithm(拉斯维加斯随机算法)

those are always correct,but its running time is a random variable.

for example:Some Supplementary Notes > Quick Sort

快速 Quick Sort: ① Runtime proportional to comparisons

② : Never compare A[i] with A[j] more than one

define $x_{ij} = \begin{cases} 1, & \text{if } i\text{th smallest one compared with } j\text{th smallest one} \\ 0, & \end{cases}$

$$E(\text{Runtime}) \le E(C \cdot \sum_{i<j} x_{ij})$$
$$= C \cdot \sum E x_{ij}$$
$$= C \cdot \sum P(i\text{th smallest}, j\text{th smallest one ever compared})$$
$$= O\left(\sum_{i<j} \frac{2}{j-i+1}\right)$$

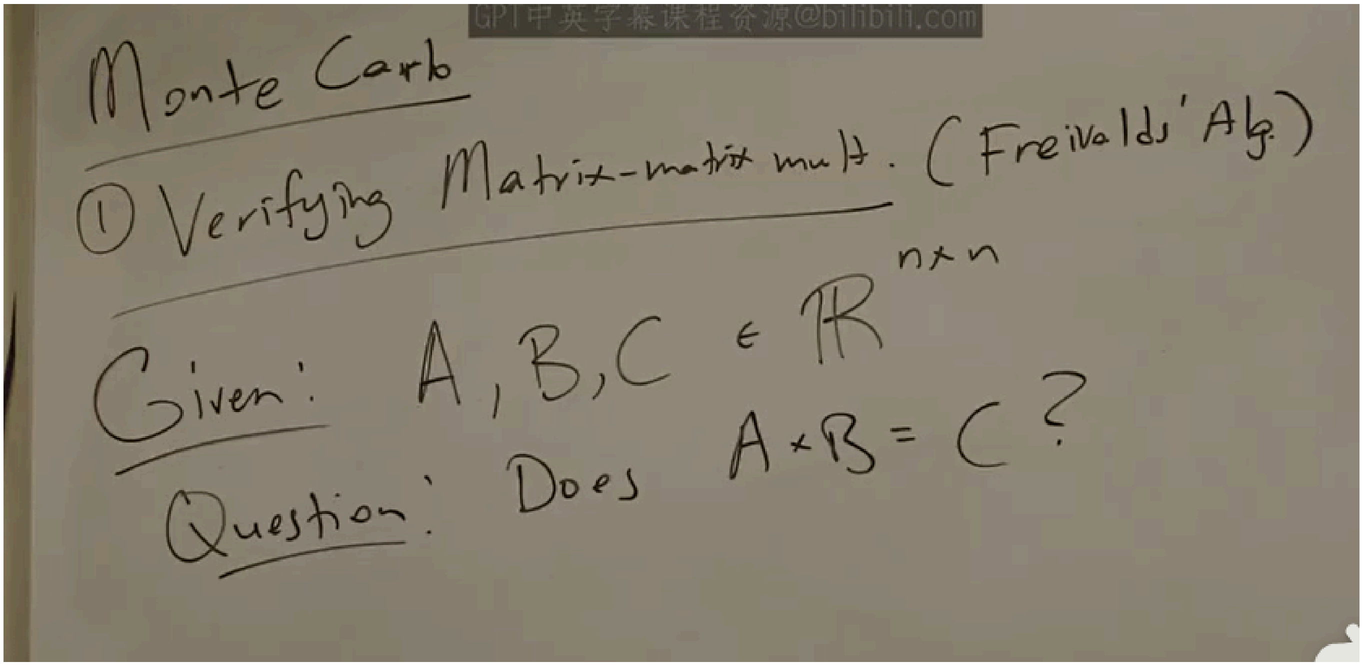
$$\sum_{i<j} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1} \cdot 2 \le 2n \cdot \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right) \le 2n \ln n$$

$$P(X \ge a) \le \frac{E(X)}{a} \qquad P(T \ge 100\, C_n \ln n) \le \frac{C n \ln n}{100 \cdot C_n \ln n} = \frac{1}{100}$$

Monte Carlo randomized algorithms(蒙特卡罗随机算法)

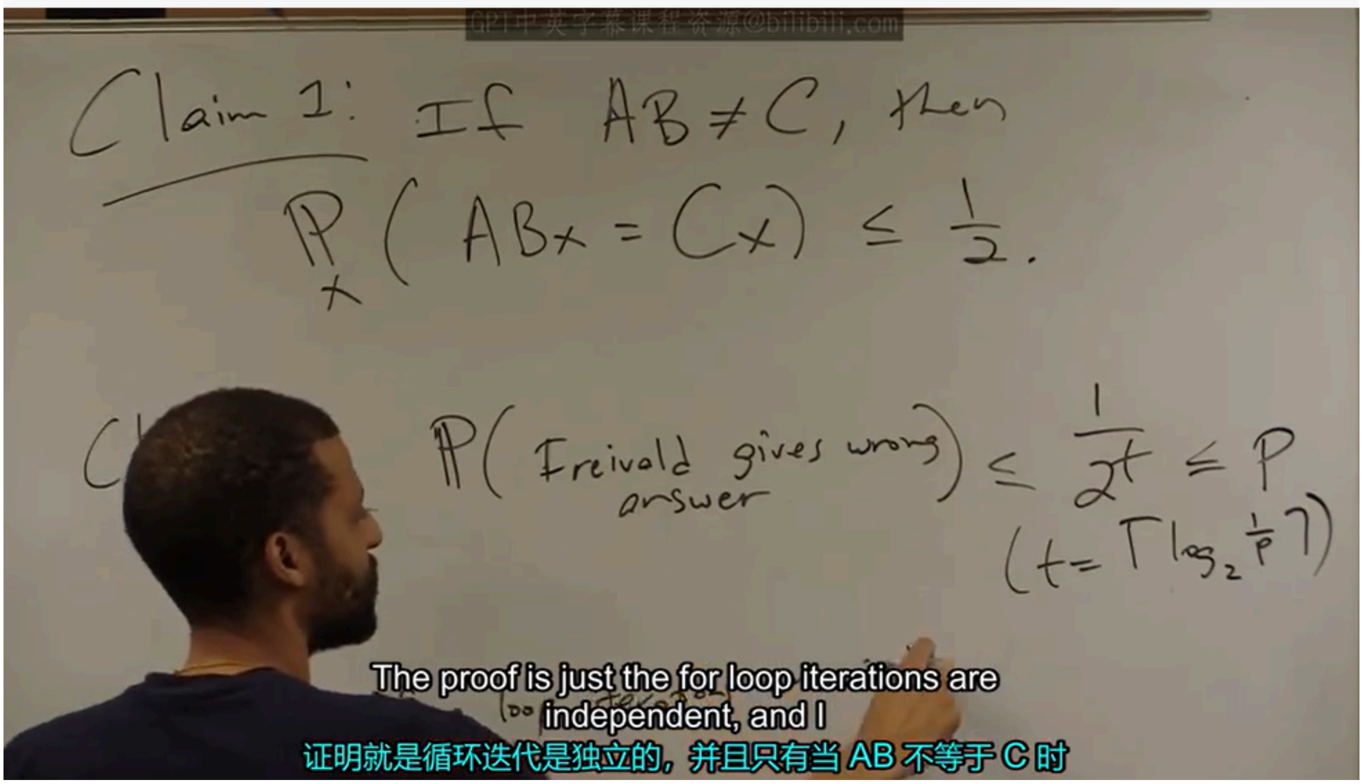
those are always fast, but its correctness is a random variable.

for example:



We can test it by create a vector X test if ABx=Cx.

So then



Claim 1: If $AB \neq C$, then
$\mathbb{P}_x(ABx = Cx) \leq \frac{1}{2}$.
$\mathbb{P}$(Freivald gives wrong answer) $\leq \frac{1}{2^t} \leq p$
$(t = \lceil \log_2 \frac{1}{p} \rceil)$
GPT中英字幕课程资源@bilibili.com
The proof is just the for loop iterations are
independent, and I
证明就是循环迭代是独立的，并且只有当 AB 不等于 C 时

Monte Carlo

① . Verifying Matrix-matrix mult

Given: $A, B, C \in R^{n \times n}$

Question: Does $A \times B = C$?

Freeval's algorithm:

① pick: $x_1, \dots, x_t \in \{0, 1\}^n$ independent

② for $i=1$ to t: only need to do three matrix vector multiplications

if $ABx_i \neq Cx_i$: return False

return true

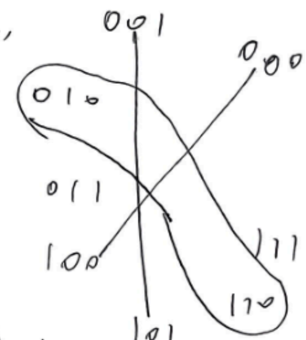
· $O(t \cdot N^2)$

x_{1j} is ~~x_{ij} trans~~ converse in "i, j"

claim 1: If $AB \neq C$, then ~~Prob~~ $P_x(ABx = Cx) \leq \frac{1}{2}$

claim2: P(Freivold gives wrong answer) $\leq \frac{1}{2^t} \leq p$

$(t = \lceil \log_2 \frac{1}{p} \rceil)$



both are good or one of each are bad

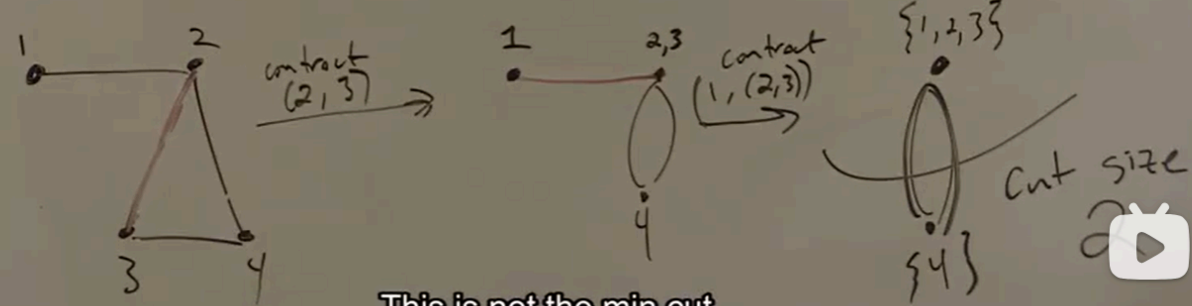
another example:

find the global mincut of the graph

We will not want to find all n-1 mincut and then find it.

So please use Karger's algorithm.

Today: will show randomized alg for case where all edge weights are 1. (Karger's contraction algorithm)



This is not the min cut.
这不是最小割。

Fix: Run Karger R times, and return smallest cut it ever finds.

$$\mathbb{P}\left(\text{Fixed Karger fails to output mincut}\right) \le \left(1 - \frac{1}{\binom{n}{2}}\right)^R$$

$$\le e^{-R/\binom{n}{2}} \quad \left(\text{fact: } \forall x \ \ 1+x \le e^x\right)$$

$$\le p \quad \left(\text{pick } R = \left\lceil \binom{n}{2} \cdot \ln \frac{1}{p} \right\rceil\right)$$

$\rightarrow$ Total time: $O(n^4 \cdot \log \frac{1}{p})$.

$$P(\text{Karger outputs mincut}) \geq \frac{1}{\binom{n}{2}}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$P(\text{fail}) \leq \left(1 - \frac{1}{\binom{n}{2}}\right)$$

run R times

$$P(\text{fail}) \leq \left(1 - \frac{1}{\binom{n}{2}}\right)^R \leq e^{-R/\binom{n}{2}} \leq p \quad \left(\text{pick } R = \binom{n}{2}\ln\frac{1}{p}\right)$$

$\Rightarrow$ Total time: $O\left(n^4 \log\frac{1}{p}\right)$

proof about claim1:

a Fixed particular mincut $(S, \bar{S})$

claim 2: $P(\text{karger outputs } (S, \bar{S})) \geq \frac{1}{\binom{n}{2}}$

notes: implies G has $\leq \binom{n}{2}$ mincuts, which is tight

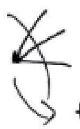
$\binom{n}{2}$ ways to do this

$P(\text{karger output } S, \bar{S}) = P(\text{for all } n-2 \text{ items, never contract any } e \in E(S, \bar{S}))$

$= \prod_{i=1}^{n-2} P(\text{don't contract } e \in E(S, \bar{S}) \mid \text{haven't contract any } e \in E(S, \bar{S}) \text{ items } 1, \dots, i-1)$

notation: mincut size $:= k$

$$= \prod_{i=1}^{n-2} \left(1 - \frac{k}{m_i}\right)$$



the degrees of it is 3

$\forall v$, degree of $v \geq k$, or there will be a cut smaller than mincut

$m_i = \frac{1}{2} \sum d_v$

$m_i \geq \frac{1}{2} k \cdot n_i = \frac{1}{2} \cdot k(n-i+1)$

$$= \prod_{i=1}^{n-2} \left(1 - \frac{k}{\frac{k}{2}(n-i+1)}\right) = \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdot \left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{3}\right)$$

$$= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdots \frac{3}{5} \frac{2}{4} \frac{1}{3}$$

$$= \frac{2}{n \cdot (n-1)} = \frac{1}{\binom{n}{2}}$$

Streaming Algorithm

introduce:
Algorithms must process a "stream" of incoming data, then be able to answer some query or family or like many different queries.
Goal:Use little memory.
in particular,I do not want to remember most of what i saw.

Chebyshev's inequality:

$\forall t > 0 \quad P(|X - E(X)| \geq t) < \dfrac{Var[X]}{t^2}$ → Variance of X

$= E((X - E(X))^2)$

$= E(X^2) - (E(X))^2$

approximate counter: ① init $n \leftarrow 0$

② $n \leftarrow n+1$

③ query: return n

naive algorithm: $\boxed{1\,1\,1\,0\,1\,1} \rightarrow \log_2 n \Rightarrow \log n$

now we'll settle for knowing n not exactly, but up to some approximate factor;

$n \leq \hat{n} \leq \alpha n$

α: approximation factor

$1 \quad \alpha \quad \alpha^2 \quad \cdots \quad \alpha^{\log_2 R}$

$\log_\alpha \log_2 n \Rightarrow O(\log \log n)$

Morris's algorithm,

init: $X \leftarrow 0$

incre...(): w.p. $\frac{1}{2^X}$, $X \leftarrow X+1$

(with probability)

o/w do nothing

query(): return $2^X - 1$

claim: $E(2^X) = n+1$

so $E(2^X - 1) = E(2^X) - 1 = n+1-1 = n$

Pf: $n=0$ $E(2^X) = 1$

Inductive step:

$$E(2^{X_{n+1}}) = \sum_{j=0}^{\infty} P(X_n = j)\left(E(2^{X_{n+1}} \mid X_n = j)\right)$$

$$= \sum_{j=0}^{\infty} P(X_n = j) \cdot \left[\frac{1}{2^j} \cdot 2^{j+1} + \left(1 - \frac{1}{2^j}\right) \cdot 2^j\right]$$

$$= \sum_{j=0}^{\infty} P(X_n = j) \cdot (2^j + 1)$$

$$= \sum_{j=0}^{\infty} P(X_n = j) + \sum_{j=0}^{\infty} P(X_n = j) \cdot 2^j$$

$$= 1 + E\,2^{X_n}$$

$$= 1 + n+1$$

$$= (n+1)+1$$

$$E\gamma = n+1$$

$$\mathrm{Var}[\gamma] = E\gamma^2 - (E\gamma)^2$$

$$= E\gamma^2 - (n+1)^2$$

$$\downarrow$$

claim: $= \frac{3}{2}n^2 + \frac{3}{2}n + 1$

$$\mathrm{Var}[\gamma] = \cancel{\tfrac{1}{2}n(n+1)}\ \frac{1}{2}\cdot n(n-1)$$

$$P(|\hat{n} - n| > \varepsilon \cdot n) \leq \frac{\mathrm{Var}[\hat{n}]}{\varepsilon^2 n^2} < \frac{\frac{1}{2}\cdot n^2}{\varepsilon^2 \cdot n^2} = \frac{1}{2\varepsilon^2}$$

But we can do better.

$\Rightarrow$ Do better: ① If A, B indep.., then $Var[A+B] = Var[A] + Var[B]$

② $Var[a \cdot Z] = a^2 \cdot Var[Z]$

Run Morris' algorithm q times in parallel, $\Rightarrow \hat{n}_1, \hat{n}_2, \hat{n}_3 \cdots, \hat{n}_q$

$$n^* = \frac{1}{q} \sum_{i=1}^{q} \hat{n}_i$$

$$E n^* = n$$

$$Var[n^*] = \sum_{i=1}^{q} Var\left[\frac{\hat{n}_i}{q}\right] = \frac{1}{q^2} \sum_{i=1}^{q} Var[\hat{n}_i] \le \frac{n^2 \cdot q}{2q^2} = \frac{n^2}{2q}$$

$$\cancel{= \sum_{j=0}^{\infty} P(X_n = j) \cdot (2^j + 1)}$$

$$\cancel{= \sum_{j=0}^{\infty} P(X_n = j) + \sum_{j=0}^{\infty} P(X_n = j) \cdot 2^j}$$

$$\cancel{= 1 + E\, 2^{X_n}}$$

$$\cancel{= 1 + n + 1}$$

$$\cancel{= (n+) + 1}$$

$$\Rightarrow P(|n^* - n| > \varepsilon n) < \frac{Var[n^*]}{\varepsilon^2 \cdot n^2}$$

$$< \frac{1}{2q \cdot \varepsilon^2}$$

设 $q = \frac{1}{2\varepsilon^2 p} \rightarrow \Downarrow$

$$\le p$$

erall: need $q = O\left(\frac{1}{\varepsilon^2 p}\right)$ copies,

Total memory: $O\left(\frac{\log\log n}{\varepsilon^2 p}\right)$

But we can do better again ! ! !

Doing better: (with probility)

①: Trivial Increment X wp 1

pro: ~~two~~ 0 Variance

con: use $O(\log n)$ memory

$$\boxed{\text{Increment } X \text{ wp } (1+n)^{X}}$$

② Morris: Increment X wp 0.5^X

Pro: $O(\log\log n)$ memory

Con: Hign Variance

Here we have another problem:we want to find the size of different numbers of a list.
For example: input 1,2,3,5,1,3 return 4
But if we store all those numbers and then compare them. it will cost a lot of time. So we use the thought of Stream Algorithm: **We just want to remember little things about the list.**

So we can treat it in this way:

Chebyshev's inequality:

init $x \leftarrow 1$

inc $x \leftarrow \min(x, h(t))$

query: $\frac{1}{x} - 1$

example: $t = 1$, Exp: $0 \quad \frac{1}{2} \quad 1 \Rightarrow \frac{1}{\frac{1}{2}} - 1 = 1$

$t = 4$, Exp: $0 \quad 1/5 \quad 2/5 \quad 3/5 \quad 4/5 \quad 1 \quad \frac{1}{1/5} - 1 = 4$

claim1 : $Ex = \frac{1}{t+1}$

Pf : $X \geq 0$ $Ex = \int_0^\infty P(X > y)\, dy$

$Ex = \int_0^1 P(X > y) dy = \int_0^1 P(\forall i = 1, \dots t, X_i > y) dy$

$= \int_0^1 (P(X_1 > y))^t dy$

$= \int_0^1 (1-y)^t dy$

$= \int_0^1 u^t du$

$= \frac{u^{t+1}}{t+1}\Big|_0^1 = \frac{1}{t+1}$

claim2 : $Ex^2 = \frac{2}{(t+1)(t+2)}$

$Ex^2 = \int_0^1 P(x > \sqrt{y}) dy$

$= \int_0^1 (1-\sqrt{y})^t dy$

$= \frac{2}{(t+1)(t+2)}$

$Var[X] = \frac{2}{(t+1)(t+2)} - \frac{1}{(t+1)^2} = \frac{t}{t+2} \cdot \frac{1}{(t+1)^2} < (Ex)^2$

run Q copies of this algorithm in parellel with independent randomness to get $X^{(1)}, \dots X^{(q)}$

set $X^\alpha = \frac{1}{q} \sum_{i=1}^{q} X^{(i)}$

query : return $\frac{1}{X^\alpha} - 1$

$P(|X^\alpha - \frac{1}{t+1}| > \frac{\varepsilon}{t+1}) \leq \frac{(t+1)^2}{\varepsilon^2} \cdot \frac{1}{(t+1)^2} \cdot \frac{1}{q} = \frac{1}{\varepsilon^2 q} < P$ (set $q = \frac{1}{\varepsilon^2 p}$)

total mem : $O(\frac{1}{\varepsilon^2 p})$

claim1: $Ex = \frac{1}{t+1}$

Pf: $X \geq 0$ $Ex = \int_0^\infty P(X > y)\,dy$

$$Ex = \int_0^1 P(X > y)\,dy = \int_0^1 P(\forall i = 1, \dots t,\ X_i > y)\,dy$$

$$= \int_0^1 (P(X_1 > y))^t\,dy$$

$$= \int_0^1 (1-y)^t\,dy$$

$$= \int_0^1 u^t\,du$$

$$= \frac{u^{t+1}}{t+1}\Big|_0^1 = \frac{1}{t+1}.$$

claim2: $Ex^2 = \frac{2}{(t+1)(t+2)}$

$$Ex^2 = \int_0^1 P(x > \sqrt{y})\,dy$$

$$= \int_0^1 (1-\sqrt{y})^t\,dy$$

$$= \frac{2}{(t+1)(t+2)}$$

$$Var[X] = \frac{2}{(t+1)(t+2)} - \frac{1}{(t+1)^2} = \frac{t}{t+2} \cdot \frac{1}{(t+1)^2} < (Ex)^2$$

run Q copies of this algorithm in parellel with independent randomness to get $X^{(1)}, \dots X^{(q)}$

set $X^\alpha = \frac{1}{q}\sum_{i=1}^{q} X^{(i)}$

query: return $\frac{1}{X^\alpha} - 1$

$$P\left(\left|X^\alpha - \frac{1}{t+1}\right| > \frac{\varepsilon}{t+1}\right) \leq \frac{(t+1)^2}{\varepsilon^2} \cdot \frac{1}{(t+1)^2} \cdot \frac{1}{q} = \frac{1}{\varepsilon^2 q} < P \quad \left(\text{set } q = \frac{1}{\varepsilon^2 P}\right)$$

total mem: $O\left(\frac{1}{\varepsilon^2 P}\right)$

and we can do it better