Agenda

What have we done?
- Probabilistic thinking!
- Balls and Bins
- Probabilistic Method
- Foundations of DTMC
- Random Walks on Graphs and Expanders

Next
- Approximate Counting and Sampling
Example 1: Number of Spanning Trees

Problem

Given $G$ connected, count the number of spanning trees.

- $A$: adjacency matrix of $G$
- $D$: diagonal matrix of vertex degrees
- $L = D - A$: Laplacian of $G$
- $L_{ij}$: submatrix of $L$ obtained by removing column $i$, row $j$
- $(-1)^{i+j} \det(L_{ij})$: $ij$-cofactor of $L$
- $0 = \mu_0 < \mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$ the Laplacian spectrum

Theorem (Matrix-Tree, also Kirchhoff’s Theorem)

Number of spanning trees of $G$ is $(-1)^{i+j} \det(L_{ij})$ for all $i, j$, which is equal to $\frac{1}{n} \mu_1 \cdots \mu_n$. 
“Dimer Covers”

Given a graph $G$, count the number of perfect matchings.

- A Pfaffian orientation of $G$ is an orientation $\vec{G}$ such that: for any two perfect matchings $M_1$ and $M_2$ of $G$, every cycle of $M_1 \cup M_2$ has an odd number of same-direction edges.

- In particular, if $\vec{G}$ is an orientation in which every even cycle is oddly oriented, then $\vec{G}$ is a Pfaffian orientation.

- Skew adjacency matrix $A_s(\vec{G}) = (a_{uv})$:

\[
a_{uv} = \begin{cases} 
+1 & (u, v) \in E(\vec{G}) \\
-1 & (v, u) \in E(\vec{G}) \\
0 & \text{otherwise}
\end{cases}
\]
Kasteleyn’s Theorem

Theorem (Kasteleyn)

For any Pfaffian orientation $\vec{G}$ of $G$,

$$\text{number of perfect matchings} = \sqrt{\det(A_s(\vec{G}))}$$

Theorem

Every planar graph has a Pfaffian orientation which can be found in polynomial time. In particular, Dimer Covers is solvable for planar graphs!

Open Question

Complexity of deciding if a graph $G$ has a Pfaffian orientation. (Known to be in $P$ if $G$ is bipartite.)
Example 1: Routing in Intermittently Connected Networks

- $G$: ad hoc network of mobile users
- For every $(u, v) \in E$, $p_{uv}$ is the probability that $u$ and $v$ are “in contact”
- For simplicity, say $p_{uv} = 1/2$
- **Want:** send a message from $s$ to $d$
- If routed through a length-$k$ $s, t$-path, delivery probability is $(1/2)^k$
- To increase delivery probability, send messages along edges of a subgraph $H \subseteq G$ such that $\text{Prob}[s$ and $t$ connected in $H]$ is maximized
- If $H = G$, we are just broadcasting $\Rightarrow$ broadcast storm problem
- If $H$ is a path, delivery prob. is too low
Routing on a Probabilistic Graph

Given $G$ (and $p_{uv}$), and a parameter $k$, find a subgraph $H \subseteq G$ with at most $k$ edges so that Prob[$s$ and $t$ connected in $H$] is maximized.

- Given $H$, how to compute Prob[$s$ and $t$ connected in $H$]? (let alone finding an optimal $H$)
- (Ghosh, Ngo, Yoon, Qiao – INFOCOM’07) The optimization problem is $\#P$-Hard, if solvable then $P = NP$
- Subtle: $P = NP$ does not necessarily imply problem solvable
Network Reliability Problem

Given $H$ (and $p_{uv}$), compute $P = NP$ and $t$ connected in $H$.

- Suppose $H$ has $m$ edges. Then, $\text{Prob}[s$ and $t$ connected in $H]$ is

$$\frac{1}{2^m} \left( \text{#subgraphs of } H \text{ which contains an } s, t\text{-paths} \right)$$

Network Reliability, Counting Version

Given $H$, find the number of subgraphs of $H$ in which there is a path from $s$ to $t$
**Example 2: \#CNF, \#DNF, 01-Perm, \#Bipartite-PM**

<table>
<thead>
<tr>
<th><strong>#CNF</strong></th>
<th>Given a CNF formula $\varphi$, count number of satisfying assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>#DNF</strong></td>
<td>Given a DNF formula $\varphi$, count number of satisfying assignments</td>
</tr>
<tr>
<td><strong>#Bipartite-PM</strong></td>
<td>Given a bipartite graph $G$, count number of perfect matchings</td>
</tr>
<tr>
<td><strong>01-Perm</strong></td>
<td>Given a 01-square matrix $A$, compute $\text{per } A$, defined by</td>
</tr>
<tr>
<td></td>
<td>$\text{per } A = \sum_{\pi \in S_n} \prod_{i=1}^{n} a_{i\pi(i)}$</td>
</tr>
</tbody>
</table>
Rough Classification of Counting Problems

“Easy” Counting Problems

- \# Subsets of a Set
- \# Spanning trees of $G$
- \# Perfect matchings in planar graphs

“Hard” Counting Problems (At least, no solution is known)

- Network reliability
- \#CNF
- \#DNF
- 01-perm, \#bipartite-PM
- \#cycles, \#Hamiltonian cycles, \#cliques, \#$k$-cliques, etc.
How to Show a Counting Problem is Hard?

Suppose we want to prove any problem $\Pi$ is “hard” to solve

Try This First

Prove that if $\Pi$ can be solved in polynomial time, then some $\mathsf{NP}$-complete problem can be solved in polynomial time.

- Typically Done with Optimization Problem.
- $\#\mathsf{CNF}$, $\#\text{HAM-CYCLES}$, ... are certainly $\mathsf{NP}$-hard
- We’ll show $\#\mathsf{DNF}$ and $\#\text{CYCLES}$ are $\mathsf{NP}$-hard to illustrate.

Try This Next

Define a new complexity class $C$ for $\Pi$, and show $\Pi$ is complete in that class. Provide evidence that $C$ is not complete as a whole.

This was what Valiant did in 1978 for $01$-$\text{PERM}$ and $\text{NETWORK RELIABILITY}$. The new class $C$ is $\#\mathsf{P}$
# DNF is $\textbf{NP}$-hard

## Theorem

*If we can count the number of satisfying assignments of a DNF formula, then we can decide if a CNF formula is satisfiable.*

Given $\varphi$ in CNF:

$$
\varphi = (x_1 \lor \bar{x}_2 \lor x_3) \land (x_2 \lor x_3 \lor \bar{x}_4)
$$

$\varphi$ is satisfiable iff $\overline{\varphi}$ has $< 2^n$ satisfying assignments.

$$
\overline{\varphi} = (\bar{x}_1 \land x_2 \land \bar{x}_3) \lor (\bar{x}_2 \land \bar{x}_3 \land x_4)
$$
Theorem

If we can count the number of cycles of a given graph in polynomial time, then we can decide if a graph has a Hamiltonian cycle in polynomial time.

To decide if \( G \) has a Hamiltonian cycle, construct \( G' \) as shown

- Each length-\( l \) cycle in \( G \) becomes \((2^m)^l\) cycles in \( G'\)
- If \( G \) has a Hamiltonian cycle, \( G' \) has at least \((2^m)^n > n^{n^2}\) cycles
- If all cycles of \( G \) have lengths \( \leq n - 1 \), there can be at most \( n^{n-1} \) cycles in \( G \), implying \( \leq (2^m)^{n-1} n^{n-1} < n^{n^2} \) cycles in \( G' \)
Sample Problems (each have a #-version)

1. **SPANNING TREE**: does $G$ have a spanning tree?
2. **BIPARTITE-PM**: does bipartite $G$ have a perfect matching?
3. **CNF**: does $\varphi$ in CNF have a satisfying assignment?
4. **DNF**: does $\varphi$ in DNF have a satisfying assignment?

- **P**: problems whose solutions can be **found efficiently**: SPANNING TREE, dnf, bipartite-PM
- **NP**: problems whose solutions can be **verified efficiently**: all four
- **FP**: problems whose solutions can be **counted efficiently**: 
  #SPANNING TREE
- **#P**: problems of **counting efficiently verifiable** solutions: all four.
A counting problem \( \#\Pi \) is \( \#\text{P} \)-complete iff it is in \( \#\text{P} \) and, if \( \#\Pi \) can be solved efficiently, then we can solve all \( \#\text{P} \) problems efficiently.

**Lemma**

\( \#\text{CNF} \) is \( \#\text{P} \)-complete (for the same reason \( \text{SAT} \) is \( \text{NP} \)-complete)

This implies \( \#\text{DNF} \) is \( \#\text{P} \)-complete. (Why?)

**Theorem**

If any \( \#\text{P} \)-complete problem can be solved in poly-time, then \( \text{P} = \text{NP} \).

The converse is not known to hold (open problem!)

**Theorem (Valiant)**

\( \#\text{BIPARTITE-PM} \) and \( \#\text{01-Perm} \) are \( \#\text{P} \)-complete
Approximate Counting: What and Why

- Suppose we want to estimate some function \( f \) on input \( x \)
  - \( x = G, f(G) = \) number of perfect matchings
  - \( x = \phi \) in DNF, \( f(\phi) = \) number of satisfying assignments

- For many problems, computing \( f(x) \) efficiently is (extremely likely to be) difficult

- The next best hope is: given \( \epsilon, \delta \), efficiently compute \( \tilde{f}(x) \) such that

\[
\text{Prob}[|\tilde{f}(x) - f(x)| > \epsilon f(x)] < \delta
\]

**Definition (FPRAS)**

A randomized algorithm producing such \( \tilde{f} \) is called a **fully polynomial time randomized approximation scheme** (FPRAS) if its running time is polynomial in \(|x|, 1/\epsilon, \log(1/\delta)\)
An Alternative Definition of FPRAS

**Definition (FPRAS)**

A fully polynomial time randomized approximation scheme (**FPRAS**) for computing $f$ is a randomized algorithm satisfying the following:

- on inputs $x$ and $\epsilon$
- $A$ outputs $\tilde{f}(x)$, such that

$$\text{Prob}[|\tilde{f}(x) - f(x)| > \epsilon f(x)] < 1/4$$

- $A$’s running time is polynomial in $|x|$ and $1/\epsilon$

The *median trick* shows the equivalence between the two definitions.
The Essence of the Monte Carlo Method

Basic idea: to estimate $\mu$

- Design an efficient process to generate $t$ i.i.d. variables $X_1, \ldots, X_t$ such that $E[X_i] = \mu$, $\text{Var}[X_i] = \sigma^2$, for all $i$ ($X_i$ is called an unbiased estimator for $\mu$)
- Output the sample mean $\tilde{\mu} = \frac{1}{t} \sum_{i=1}^{t} X_i$
- Chebyshev gives the following theorem

**Theorem (Unbiased Estimator Theorem)**

If $t \geq \frac{4\sigma^2}{\epsilon^2 \mu^2}$, then

$$\text{Prob}[|\tilde{\mu} - \mu| > \epsilon \mu] < \frac{1}{4}.$$ 

In particular, if $X_i$ are all indicators, then $\sigma^2 = \mu(1 - \mu) \leq \mu$; we only need $t \geq \frac{4}{\epsilon^2 \mu}$. 

©Hung Q. Ngo (SUNY at Buffalo) 
CSE 694 – A Fun Course
Potential Bottlenecks of the Monte Carlo Method

- Each single sample value \( X_i \) must be generated efficiently.
- The number of samples \( t \) needs to be a polynomial in \( |x| \) (and \( 1/\epsilon \)).
- So, if \( \mu \) is really small then we’re in trouble!
# DNF with Naive Monte Carlo Algorithm

**Line of thought**

- \( f = f(\varphi) \) is the number of satisfying assignments.
- Probability that a random truth assignment satisfies \( \varphi \) is \( \mu = f / 2^n \).
- Let \( X_i \) indicates if the \( i \)th truth assignment satisfies \( \varphi \).
- \( \text{Prob}[X_i = 1] = \mathbb{E}[X_i] = \mu \).
- After taking \( t \) samples, output

\[
\tilde{f} = 2^n \tilde{\mu} = 2^n \cdot \frac{1}{t} \sum_{i=1}^{t} X_i
\]

- Then, by the unbiased estimator theorem, when \( t \geq \frac{4}{\epsilon^2 \mu} \) we have

\[
\text{Prob}[|\tilde{f} - f| > \epsilon f] = \text{Prob}[|\tilde{\mu} - \mu| > \epsilon \mu] < 1/4
\]

- If \( f \ll 2^n \), say \( f = n^2 \), then \( \mu = n^2 / 2^n \) and \( t = \Omega(2^n / n^2) \).
What is the Main Problem with the Naive Method?

- To find a few needles in a haystack, we need **many** samples.
- More concretely, the sample space is too large, while the “good set” is too small.
- Karp-Luby (STOC 1973) designed a much smaller sample space from which we can still sample efficiently.
The Karp-Luby Algorithm for $\#\text{DNF}$

- Suppose $\varphi$ has $m$ terms

$$\varphi = T_1 \lor T_2 \lor \cdots \lor T_m = (\overline{x}_1 \land x_2 \land \overline{x}_3) \lor (\overline{x}_2 \land x_4) \lor \cdots$$

- Let $S_j$ be the set of assignments satisfying $T_j$ which has $v_j$ variables

Then, $|S_j| = 2^{n-v_j}$; and we want $f = \left| \bigcup_{j=1}^{\pi} S_j \right|$.

- The haystack

$$\Omega = \left\{ (a, j) \mid a \in S_j \right\}$$

$$|\Omega| = \sum_{j=1}^{m} 2^{n-v_j} \leq \frac{m}{2^n}$$

- The needles (represent each satisfying $a$ by the minimum $j$ for which $a \in S_j$)

$$N = \left\{ (a, j) \mid j = \min(j', (a, j') \in \Omega) \right\}, \implies f = |N|$$
The Karp-Luby Algorithm for #DNF

The Algorithm

for \( i = 1 \) to \( t \) do
  Choose \((a, j)\) uniformly from \( \Omega \)
  \[ X_i = \begin{cases} 1 & (a, j) \in N \\ 0 & \text{otherwise} \end{cases} \]
end for

Output \(|\Omega| \cdot \frac{1}{t} \sum_{i=1}^{t} X_i\)

The Analysis

- \( \text{Prob}[X_i = 1] = E[X_i] = \frac{|N|}{|\Omega|} \)
- To chose \((a, j)\) uniformly from \( \Omega \), pick \( j \) with probability \( \frac{|S_j|}{\sum |S_j|} \), then choose \( a \in S_j \) uniformly
- Checking if \((a, j) \in N\) is the same as checking if \( a \) satisfies \( T_{j'} \) for some \( j' < j \).
The algorithm can be used to estimate

\[
\left| \bigcup_{j=1}^{m} S_j \right|
\]

for any collection of sets \( S_j \) for which similar operations can be done efficiently.
Almost Uniform Sampling

Definition (FPAUS)

A **fully polynomial time almost uniform sampler** is a randomized algorithm $A$:

- $A$’s input is an instance $x$ of the problem (like a graph $G$)
- $A$ internally chooses a random string $R$
- $A$ outputs $A(x, R) \in \Omega$, $\Omega$ is the set of solutions to $x$
- the **total variation distance** between $A$’s output distribution and the uniform distribution is at most $\epsilon$

$$\max_{S \subseteq \Omega} \left| \frac{\Pr[R \in S]}{|\Omega|} - \frac{|S|}{|\Omega|} \right| \leq \epsilon$$

- $A$’s running time is polynomial in $|x|$ and $\log(1/\epsilon)$
(Approximate) Sampling and Counting

<table>
<thead>
<tr>
<th>Exact Counting</th>
<th>$\rightarrow$</th>
<th>Exact Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\downarrow$</td>
<td></td>
<td>$\downarrow$</td>
</tr>
</tbody>
</table>

| Approximate Counting (FPRAS) | $\leftrightarrow$ | Approximate Sampling (FPAUS) |

(* means “true for a class of problems,” which is fairly large)
Approximate Sampling $\implies$ Approximate Counting

Counting number of matchings ($\#\text{MATCHINGS}$): given a graph $G$
- $\mathcal{M}(G)$ = set of matchings (not necessarily perfect)
- $f(G) = |\mathcal{M}(G)|$
- Compute $f(G)$

**Theorem**

*If there is a FPAUS for $\#\text{MATCHINGS}$ then there is a FPRAS for it too*
Making Use of “Self-Reducibility”

- Suppose $G = (V, \{e_1, e_2, \ldots, e_m\}$
- Let $G_k = (V, \{e_1, \ldots, e_k\})$, $0 \leq k \leq m$
- Key idea:

\[
f(G) = f(G_m) = \frac{f(G_m)}{f(G_{m-1})} \cdot \frac{f(G_{m-1})}{f(G_{m-2})} \cdots \frac{f(G_1)}{f(G_0)} \cdot f(G_0) = \frac{1}{r_m} \cdot \frac{1}{r_{m-1}} \cdots \frac{1}{r_1} \cdot 1
\]

We will approximate all the

\[
r_k = \frac{f(G_{k-1})}{f(G_k)}, \quad 1 \leq k \leq m
\]

then take the reciprocal of their product as an estimate for $f(G)$
How Well Must We Approximate the $r_k$?

- Suppose $\tilde{r}_k$ is an $(\bar{\epsilon}, \bar{\delta})$-approximation for $r_k$, $1 \leq k \leq m$
- Want: $\tilde{f} = \frac{1}{\tilde{r}_1 \cdots \tilde{r}_m}$ to be an $(\epsilon, \delta)$-approximation for $f = \frac{1}{r_1 \cdots r_m}$:

$$\text{Prob} \left[ \left| \frac{1}{\tilde{r}_1 \cdots \tilde{r}_m} - \frac{1}{r_1 \cdots r_m} \right| < \epsilon \frac{1}{r_1 \cdots r_m} \right] > 1 - \delta$$

which is the same as

$$\text{Prob} \left[ 1 - \epsilon < \frac{r_1 \cdots r_m}{\tilde{r}_1 \cdots \tilde{r}_m} < 1 + \epsilon \right] > 1 - \delta$$

- What we have is:

$$\text{Prob} \left[ |\tilde{r}_k - r_k| < \bar{\epsilon} r_k \right] > 1 - \bar{\delta}$$

which is equivalent to

$$\text{Prob} \left[ \left(1 + \bar{\epsilon}\right)^{-1} < \frac{r_k}{\tilde{r}_k} < \left(1 - \bar{\epsilon}\right)^{-1} \right] > 1 - \bar{\delta}$$
Choose $\bar{\delta} = \delta/m$, then

$$\text{Prob} \left[ (1 + \bar{\epsilon})^{-1} < \frac{r_k}{\tilde{r}_k} < (1 - \bar{\epsilon})^{-1}, \text{ for all } k \right] > 1 - \delta$$

Hence,

$$\text{Prob} \left[ (1 + \bar{\epsilon})^{-m} < \prod_{k=1}^{m} \frac{r_k}{\tilde{r}_k} < (1 - \bar{\epsilon})^{-m} \right] > 1 - \delta$$

Now, setting $\bar{\epsilon} = \frac{\epsilon}{4m}$ we get

$$(1 + \bar{\epsilon})^{-m} \geq 1 - \epsilon$$
$$(1 - \bar{\epsilon})^{-m} \leq 1 + \epsilon$$
We made use of a subset of the following inequalities:

\[
\begin{align*}
1 - x & \leq e^{-x} \quad \forall x \in [0, 1] \\
1 - x & > e^{-x} - x^2 \quad \forall x < 1 \\
1 - x & > e^{-x} - \frac{1}{2} x^2 - \frac{1}{2} x^3 \quad \forall x < 1 \\
1 + x & \leq e^x \quad \forall x \in [-1, 1] \\
1 + x & > e^x - \frac{1}{2} x^2 \quad \forall x > -1 \\
1 + x & > e^x - \frac{1}{2} x^2 + \frac{1}{4} x^3 \quad \forall x > -1
\end{align*}
\]
Estimating $r_k$: Which Needles? In Which Haystack?

To estimate $r_k = \frac{f(G_{k-1})}{f(G_k)}$:

- **The haystack:** $\Omega_k = \mathcal{M}(G_k)$
- **The needles:** $\Omega_{k-1} = \mathcal{M}(G_{k-1})$
- Are there enough needles to reduce number of samples? **yes!**

$$r_k \geq \frac{1}{2}$$

- Thus, if we had an *exact* uniform sampler we only need $t \geq \frac{4}{\bar{\epsilon}^2 r_k}$ samples to get an $(\bar{\epsilon}, 1/4)$-approximation for $r_k$

Main Question Now

How many samples does an $(\bar{\epsilon}, 1/4)$-approximator for $r_k$ need if it only has access to a FPAUS, i.e. it can only sample approximately uniformly from $\Omega_k$?
Number of Samples from a FPAUS

The Algorithm

- Let $A$ be an $\epsilon'$-FPAUS for $\Omega_k$ ($\epsilon'$ to be determined)
- Take $t$ samples using $A$, let $X_i$ indicate if the $i$th sample $\in \Omega_{k-1}$
- Output $\tilde{r}_k = \frac{1}{t} \sum_{i=1}^{t} X_i$ as an estimate for $r_k$

The Analysis

- **Want** $\text{Prob}[|\tilde{r}_k - r_k| > \bar{\epsilon}r_k] < 1/4$, in other words,
  $$\text{Prob}[r_k - \epsilon r_k \leq \tilde{r}_k \leq r_k + \epsilon r_k] \geq 3/4$$

- **What do we know?**
  - From definition of $A$, $\text{Prob}[X_i = 1]$ is near $r_k$
  - Thus, $E[\tilde{r}_k]$ is near $r_k$ (within $\epsilon'$)
  - $\tilde{r}_k$ is near $E[\tilde{r}_k]$ with high probability if $t$ is sufficiently large (why?)
  - Should be able to get what we want from here
The analysis, more precisely:

- By definition of $A$,

$$r_k - \epsilon' \leq \text{Prob}[X_i = 1] = E[X_i] \leq r_k + \epsilon'$$

Thus,

$$r_k - \epsilon' \leq E[\tilde{r}_k] \leq r_k + \epsilon'$$

- To apply Chebyshev, need

$$\text{Var}[\tilde{r}_k] = \frac{1}{t^2} \sum_{i=1}^{t} \text{Var}[X_i] \leq \frac{1}{t} E[\tilde{r}_k]$$

Thus, by Chebyshev

$$\text{Prob}[|\tilde{r}_k - E[\tilde{r}_k]| > a E[\tilde{r}_k]] < \frac{\text{Var}[\tilde{r}_k]}{a^2 (E[\tilde{r}_k])^2} \leq \frac{1}{ta^2 E[\tilde{r}_k]}$$
Number of Samples from a FPAUS

- Since $\mathbb{E}[\tilde{r}_k] \geq r_k - \epsilon' \geq 1/3$

  \[
  \text{Prob}\left[(1 - a)\mathbb{E}[\tilde{r}_k] \leq \tilde{r}_k \leq (1 + a)\mathbb{E}[\tilde{r}_k]\right] \geq 1 - \frac{1}{ta^2}\mathbb{E}[\tilde{r}_k] \geq 1 - \frac{3}{ta^2} \geq 3/4
  \]

  if we take $t \geq \frac{12}{a^2}$ samples.

- Putting things together

  \[
  \text{Prob}\left[(1 - a)(r_k - \epsilon') \leq \tilde{r}_k \leq (1 + a)(r_k + \epsilon')\right] \geq 3/4
  \]

- Now, just need to choose $a$ and $\epsilon'$ so that

  \[
  (1 - a)(r_k - \epsilon') \geq (r_k - \bar{\epsilon}r_k) \\
  (1 + a)(r_k + \epsilon') \leq (r_k + \bar{\epsilon}r_k)
  \]

- $a = \bar{\epsilon}/4$ and $\epsilon' = \bar{\epsilon}/8$ work!
To Summarize

To get \((\epsilon, \delta)\)-approximation for \(f\), need

- \((\bar{\epsilon}, \bar{\delta})\)-approximation for each \(r_k\), where \(\bar{\epsilon} = \epsilon / 4m\) and \(\bar{\delta} = \delta / m\)

To get \((\bar{\epsilon}, \bar{\delta})\)-approximation for \(r_k\), need

- \(\epsilon'\)-FPAUS for \(\Omega_k\), with \(\epsilon' = \bar{\epsilon} / 8 = \epsilon / (64m)\)
- this many samples:

\[
\frac{12}{a^2} \Theta \left( \log \left( \frac{1}{\bar{\delta}} \right) \right) = \frac{192}{\bar{\epsilon}^2} \Theta \left( \log \left( \frac{m}{\delta} \right) \right) = \frac{3072m^2}{\epsilon^2} \Theta \left( \log \left( \frac{m}{\delta} \right) \right)
\]

In total, we invoke the FPAUS \(\frac{3072m^3}{\epsilon^2} \Theta \left( \log \left( \frac{m}{\delta} \right) \right)\) times.

(Number of invocations can be reduced to \(\tilde{O}(m^2)\) with a cleverer application of Chebyshev)