CS5234 Cheat Sheet

Gabriel Yeo

Inequalities

Markov: $P(X \geq \alpha) \leq \frac{E(X)}{\alpha}$

Chebyshev: $P(|X-\mu| \geq k) \leq \frac{Var(X)}{k^2}$

Chernoff: $P(X \leq (1-\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2}} \leq e^{-\frac{\delta^2\mu}{3}}$

$P(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{2+\delta}} \leq e^{-\frac{\delta^2\mu}{3}}$

$X = X_1 + ... + X_n$,

$X_i \in \{0,1\}$, (ind. Bernoulli)

Chernoff: $P(|X-\mu| \geq \delta\mu) \leq 2e^{-\frac{\delta^2\mu}{3}}$

Hoeffding: $P(|X-\mu| \geq t) \leq 2e^{\frac{-2t^2}{n}}$

$X = X_1 + ... + X_n$,

$X_i \in [0,1]$

Hoeffding: $P(|\overline{X} - E[\overline{X}]| \geq t) \leq 2e^{-2nt^2}$

$\overline{X} = \frac{1}{n}(X_1 + ... + X_n)$

Hoeffding (gen): $P(|\overline{X} - E[\overline{X}]| \geq t) \leq 2e^{-\frac{2t^2}{\sum(a_i-b_i)^2}}$

$X_i \in [a_i, b_i]$

Facts

Taylor expansion: $e^x = 1 + x + \frac{x^2}{2!} + ... = \sum^{\infty} \frac{x^n}{n!}$

For $0 < x \leq 1$: $1 - x \leq e^{-x} \leq 1 - \frac{x}{2}$

For $0 < x \leq 1$: $\frac{1}{e^2} \leq (1-x)^{1/x} \leq \frac{1}{e}$

Approx $\binom{a}{b}$: $(\frac{a}{b})^b \leq \binom{a}{b} \leq (\frac{ea}{b})^b$

For $0 < x < 1$: $\sum_{i=0}^{\infty} a^i = \frac{1}{1-a}$

Natural log: $ln(n-1) \leq \sum_{i=1}^{n} \frac{1}{i} \leq ln(n)+1$

Powers of 2: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

$1/e$: $= 0.36787944117$

$1/e^2$: $= 0.13533528323$

Probability

Expectation: $E[X] = \sum_{v \in D} v \cdot Pr[X = v]$

Variance: $Var[X] = E[(X - E[X])^2)]$

$= E[X^2] - E[X]$

More variance: $Var[aX] = a^2 Var[X]$

Linearity of expectation: $E[\sum X_i] = \sum E[X_i]$

If independent X_i: $Var[\sum X_i] = \sum Var[X_i]$

Conditional: $P[XandY] = P[X|Y] \cdot P[Y]$

Independence: $P[XandY] = P[X] \cdot P[Y]$

Union bound: $P[\bigcup E_i] \leq \sum P[E_i]$

Complexities

Algorithm	Classical	Approx
BFS	$O(n+m)$	
Connected?	$O(n+m)$	$O(\frac{1}{\epsilon^2 d})$
# CCs	$O(n+m)$	$O(\frac{d}{\epsilon^3})$
MST weight	$O(mlogn)$	$O(\frac{dW^4 logW}{\epsilon^3})$
Prim's	$O((m+n) \cdot logn)$	
Prim's (Fib heap)	$O(m + nlogn)$	
Kruskal's (MST)	$O(mlogn)$	
Dijkstra	$O((m+n) \cdot logn)$	
Dijkstra (Fib heap)	$O(m + nlogn)$	

Lecture 1 All 0s

Give an array A with n elements, $A[i] \in 0, 1$:
(1) If array has $\geq \epsilon n$ 1's, return False with probability at least $1 - \delta$:
Assume $\geq \epsilon n$ 1's, then for sample i:
$Pr[A[i] = 1] \geq \epsilon n/n \geq \epsilon$.

$$\begin{aligned} Pr(\text{all samples are 0}) &\leq (1-\epsilon)^s \\ &\leq (1-\epsilon)^{\frac{ln(1/\delta)}{\epsilon}} \\ &\leq e^{-ln(1/\delta)} \\ &\leq \delta \end{aligned}$$

Fix $s = \frac{ln(1/\delta)}{\epsilon}$

Lecture 1 Number of 1s

Give an array A with n elements, $A[i] \in 0, 1$:
(1) Find number of 1's $\pm\epsilon$ with probability at least $1 - \delta$.
Let Y_i be s independent samples in $[0, 1]$.
Output $= Z = 1/s \sum Y_i$
Probability of failure:

$$\begin{aligned} Pr(|Z - E[Z]| \geq \epsilon) &\leq 2e^{-2s\epsilon^2} \\ &\leq 2e^{-2\frac{ln(2/\delta)}{2\epsilon^2}\epsilon^2} \\ &\leq 2e^{-ln(2/\delta)} \\ &\leq 2e^{ln(\delta/2)} \\ &\leq 2 \cdot \delta/2 \\ &\leq \delta \end{aligned}$$

Fix $s = \frac{ln(2/\delta)}{2\epsilon^2}$

Lecture 2 Connectivity

G is ϵ-close to connected if you can modify at most ϵnd entries in the adjaceny list to make it connected. If $G(V, E)$ is connected, output True, else if ϵ-far from connected, output False.

```
Connected(G, n, d, ε)
    Repeat 16/εd times:
        - Choose a random node u
        - Do a BFS from u, stopping after 8/εd nodes
          seen.
        - If CC of u has ≤ 8/εd nodes, return FALSE.
    return TRUE
```

BFS cost $= \frac{8}{\epsilon d} \cdot d$
Total complexity $= O(\frac{1}{\epsilon^2 d})$

Lecture 2 Connected Components

Output CC such that: $|CC(G) - C| \leq \epsilon n$, w.p. $> 1 - \delta$
Define $n(u)$ = num nodes in the CC of u.
$cost(u) = 1/n(u)$.
$\sum_{u \in CC_i} cost(u) = 1$.

```
sum = 0
for j = 1 to s:
    - Sample u randomly
    - BFS from u, stop when see up to 2/ε nodes
    - If BFS found > 2/ε node:
        - sum := sum + ε/2
    - Else:
        - sum := sum + cost(u)
return n · (sum/s)
```

Let $\overline{C} = \sum cost(u_j)$
Let $Y_j = cost(u_j)$ of our sample j
$|CC(G) - \overline{C}| \leq \epsilon n/2$
$E[Y_j] = \sum \frac{1}{n} cost(u_i) = \frac{1}{n}\overline{C}$
$E[\sum Y_j] = s \cdot E[Y_j] = \frac{s}{n}\overline{C}$
Since we output $\frac{n}{s}\sum Y_j$, we get $E[\frac{n}{s}\sum Y_j] = \overline{C}$

$$\begin{aligned} &P(|\overline{C} - \frac{n}{s}\sum Y_j| > \epsilon n/2) \\ &= P(|E[\sum Y_j] - \sum Y_j| > \frac{s}{n}\epsilon n/2) \\ &= P(|E[\sum Y_j] - \sum Y_j| > s\epsilon/2) \\ &\leq 2e^{-2\epsilon^2 s^2/4s} \\ &\leq 2e^{-\epsilon^2 s/2} \leq \delta \end{aligned}$$

Set $s = \frac{2}{\epsilon^2} ln(2/\delta)$.
Complexity $= 2d/\epsilon \cdot O(\frac{1}{\epsilon^2} ln(1/\delta)) = O(\frac{d}{\epsilon^3} ln(1/\delta))$.

Lecture 3 MST weight

Output M such that: $M = MST(G)(1 \pm \epsilon)$ w.p. $> 1-\delta$.
Let G_j be the graph with edge weights j and below.
Let C_j be the connected components in G_j.
$MST(G)$ contains $C_j - 1$ edges of weight $> j$.
Therefore:

$$MST(G) = (n - C_1) + \sum_{j-1}^{W-1} (j+1)(C_j - C_j + 1)$$
$$= n - W + \sum_{j-1}^{W-1} C_j$$

```
sum := n - W
for j = 1 to W - 1:
    - X_j = ApproxCC(G_j, d, ε'δ')
    - sum := sum + X_j
return sum
```

Sum of errors: $(\epsilon' n)(W-1)$
Set $\epsilon' = \epsilon/W$, then sum of errors $\leq \epsilon n$.
Set $\delta' = \delta/W$. Then $P(anyfail) \leq \sum_{1}^{W-1} \delta/W \leq \delta$.
$MST(G) - \epsilon n \leq sum \leq MST(G) + \epsilon n$
Since $MST(G) \geq n-1 \geq n/2$, $n \leq 2MST(G)$,
$MST(G)(1-2\epsilon) \leq sum \leq MST(G)(1+2\epsilon)$

Lecture 3 Maximal Matching

Return the size of the Maximal Matching.

```
query(e):
    for all neighbours e' of e:
        if hash(e') < hash(e)
            if query(e') = TRUE
                return FALSE
    return TRUE

sum := 0
for j = 1 to s:
    - Choose edge e uniformly at random.
    - if (query(e) = True)
        sum := sum + 1
return m·sum/s
```

$E[cost] = 2\sum_{k=1}^{\infty} \frac{d^k}{k!} = O(e^d)$.
$sum = MM(G) \pm \epsilon m$.
Complexity $= O(\frac{e^d}{\epsilon^2})$.
Can do better: $O(\frac{d^4}{\epsilon^2})$, even $O(\frac{d^2}{\epsilon^2})$, and reduce error to $\pm\epsilon n$.

Lecture 3 Yao's Mini-Max

Every randomized algorithm on a worst-case input is always slower than the best deterministic algorithm on the worst distribution.

$$\forall A \in R : \max_{x \in X}(E[cost(A,x)])$$
$$\geq \min_{B \in D}(E[cost(B, xchosenfrom\gamma)])$$

Recipe:

- Choose distribution γ.
- Show that the expected cost of every deterministic algorithm from γ is greater than some cost c.
- Conclude that every randomized algorithm has at least one input with expected cost at least as bad as c.

Lecture 4 Misra Gries

$count(x)$: $N(x) - \epsilon m \leq count(x) \leq N(x) + \epsilon m$.
Heavy Hitters: returns

- every item that appears $\geq 2\epsilon m$ times.
- no item that appears $< \epsilon m$ times.

```
Set P of <item, count> pairs
For each u in stream S:
    1. if <u, c> is in set P, increment c.
    2. else add <u, 1> to set P.
    3. if |P| > k, decrement count c for each item.
    4. Remove all items from P with count c = 0.

Count(x):
    1. if <x, c> is in P, return c.
    2. else return 0.
```

Choose $k = 1/\epsilon$. Then $N(x) - \epsilon m \leq count(x) \leq N(x)$.
Space required: $O(klogm)$.
Proof:

- Count of x is incremented $N(x)$ times in total.
- Total increments is m.
- When $count(x)$ is decremented, at least k items are also decremented.
- At most m decrements in total.
- So $count(x)$ is decremented at most m/k times.

For Heavy Hitters: return x if $count(x) \geq \epsilon m$.

Lecture 7 External Memory Model

Cache size $= M$. Block size $= B$. Number of lines (cache slots) $= M/B$.
Assumptions:

- One cache level.
- Only memory access has cost.
- Ideal cache and replacement.

Problem	Classical	EMM
Scan Linked List	$O(N)$	$O(N)$
Scan Array	$O(N)$	$O(N/B)$
Search Linked List	$O(N)$	$O(N)$
Search Red-black tree	$O(logN)$	$O(logN)$
Search Array	$O(logN)$	$O(log(N/B))$
Search B-tree	$O(log_B N)$	$O(log_B N)$
Sort B-tree	$O(NlogN)$	$O(Nlog_B N)$
Read/Write B-tree	$O(log_B N)$	$O(log_B N)$

Lecture 7 Caching

$(a,b)-tree$ with n keys has height $\leq log_a(\frac{n}{a}) + 1$.
At most n/a leaves. Every node except the root has degree $> a$.
Node at height $log_a(\frac{n}{a})$ has $\geq a^{log_a(\frac{a}{n})} \geq \frac{n}{a}$ leaves.
Corollary: if $a \geq B$, then $(a,b)-tree$ with n keys has height $O(log_B n)$.
Amortized cost of split/share/merge is $O(1/B)$, thus $O((1/B)log_B N)$ per operation.
With parent points: insert may cost $O(Blog_B N)$ if every level needs to split.