

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:	$P(X \leq (1 - \delta)\mu) \leq e^{-\frac{\delta^2 \mu}{2}} \leq e^{-\frac{\delta^2 \mu}{3}}$ $P(X \geq (1 + \delta)\mu) \leq e^{-\frac{\delta^2 \mu}{2 + \delta}} \leq e^{-\frac{\delta^2 \mu}{3}}$ $X = X_1 + \dots + X_n$, $X_i \in \{0, 1\}$, (ind. Bernoulli)
Chernoff:	$P(X - \mu \geq \delta\mu) \leq 2e^{-\frac{\delta^2 \mu}{3}}$
Hoeffding:	$P(X - \mu \geq t) \leq 2e^{-\frac{2t^2}{n}}$ $X = X_1 + \dots + X_n$, $X_i \in [0, 1]$
Hoeffding:	$P(\bar{X} - E[\bar{X}] \geq t) \leq 2e^{-2nt^2}$ $\bar{X} = \frac{1}{n}(X_1 + \dots + X_n)$
Hoeffding (gen):	$P(\bar{X} - E[\bar{X}] \geq t) \leq 2e^{-\frac{2t^2}{\sum (a_i - b_i)^2}}$ $X_i \in [a_i, b_i]$

Facts

Taylor expansion:	$e^x = 1 + x + \frac{x^2}{2!} + \dots = \sum \frac{x^n}{n!}$
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 $\binom{a}{b}$:	$\left(\frac{a}{b}\right)^b \leq \binom{a}{b} \leq \left(\frac{ea}{b}\right)^b$
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:	$\sum_{i=0}^n 2^i = 2^{n+1} - 1$
$1/e$:	$= 0.36787944117$
$1/e^2$:	$= 0.13533528323 \leq 1/6$
$1/e^3$:	$= 0.04978706836 \leq 1/20$
Variance:	$Var[X] = E[(X - E[X])^2]$ $= E[X^2] - E[X]^2$

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	$O(m \log n)$	$O(\frac{dW^4 \log W}{\epsilon^3})$
Prim's	$O((m + n) \cdot \log n)$	
Prim's (Fib heap)	$O(m + n \log n)$	
Kruskal's (MST)	$O(m \log n)$	
Dijkstra	$O((m + n) \cdot \log n)$	
Dijkstra (Fib heap)	$O(m + n \log n)$	

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:

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

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

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

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

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 2 Connectivity

G is ϵ -close to connected if you can modify at most ϵnd entries in the adjacency 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 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:

$$\begin{aligned}
 MST(G) &= (n - C_1) + \sum_{j=1}^{W-1} (j+1)(C_j - C_{j+1}) \\
 &= n - W + \sum_{j=1}^{W-1} C_j
 \end{aligned}$$

```

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 $P(\text{anyfail}) \leq \sum_{j=1}^{W-1} \delta/W \leq \delta$.
 $MST(G) - \epsilon n \leq \text{sum} \leq MST(G) + \epsilon n$
 Since $MST(G) \geq n - 1 \geq n/2$, $n \leq 2MST(G)$,
 $MST(G)(1 - 2\epsilon) \leq \text{sum} \leq MST(G)(1 + 2\epsilon)$

Lecture 5 Spanner

If graph H has $girth(H) > 2k \rightarrow H$ has $O(n^{1+1/k})$ edges.
 $\log(n)$ -spanner space with $k = \log(n)$: requires $O(n \log n)$ space.

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

```

$$E[cost] = 2 \sum_{k=1}^{\infty} \frac{d^k}{k!} = O(e^d).$$

$$sum = MM(G) \pm \epsilon m.$$

$$\text{Complexity} = O\left(\frac{e^d}{\epsilon^2}\right).$$

Can do better: $O\left(\frac{d^4}{\epsilon^2}\right)$, even $O\left(\frac{d^2}{\epsilon^2}\right)$, 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.

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

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: $O(k \log m)$.

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 6 k-Median Clustering

Cost of each node i : $C_i = \sum_j x_{i,j} d(p_i, p_j)$.

The LP minimizes $\min \sum_i C_i$.

Goal: round fractional LP such that $C'_i \leq 4C_i$.

- If some p_i is within $4C_j$ of p_j , remove p_i from centers.
- In other words: if there is q s.t. $d(p_i, q) \leq 2C_i$ and $d(p_j, q) \leq 2C_j$, delete p_i .
- $C_j \leq C_i$ because of the order of node processing.

Goal: less than $2k$ centers.

- $\sum_{i: d(p_i, p_j) \leq 2C_j} y_i \geq 1/2$
- Since y 's sum to k , if $V(i)$ are disjoint, cannot add more than $2k$ points to S .

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	EMM
Scan Array	$O(N/B)$
Search Array	$O(\log(N/B))$
Sort B-tree	$O(N \log_B N)$
Search B-tree	$O(\log_B N)$
Insert/Delete B-tree	$O(\log_B N)$
Search BufferTree	$O(\log N)$
Insert BufferTree	$O((1/B) \log N)$
Ex.MergeSort/BufferTree	$O(\frac{N}{B} \log_{M/B} N/B)$
BFS	$O(n + \frac{n}{B} \log_{M/B} m/B)$

Lecture 7 Caching

B-Trees: (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 $\geq a^{\log_a(\frac{n}{a})} \geq \frac{n}{a}$ leaves.

Corollary: if $a \geq B$, then (a, b) -tree with n keys has height $O(\log_B n)$.

BufferTrees: Amortized cost of split/share/merge is $O(1/B)$, thus $O((1/B) \log_B N)$ per operation.

With parent pointers: insert may cost $O(B \log_B N)$ if every level needs to split.

Lecture 9: van Emde Boas Tree

Cache-oblivious structure for searching.

- Let T_r be the subtree consisting of all the nodes of depth $\leq \lfloor (\log n)/2 \rfloor$.
- Let T_2, T_2, \dots, T_k be the subtrees consisting of all the nodes of depth $> \lfloor (\log n)/2 \rfloor$, each subtree rooted at a unique node of depth $\lfloor (\log n)/2 \rfloor + 1$.
- Then arrange the array as follows: $L(T_r), L(T_1), L(T_2), \dots, L(T_k)$.

Single value search: $O(\log_B n)$.

Range search with k values in range $[k_1, k_2]$: $O(\log_B n + k/B + 1)$.

Lecture 10: BFS in the EMM

We can BFS in $O(n + \frac{m}{B} \log_{M/B} m/B)$.

Inductive invariant: L_i and L_{i-1} are sorted.

Let $N(L_i)$ be the set of edges adjacent to nodes in L_i .

1. **Enumeration:** Iterate through each $u \in L_i$ to make L_{i+1}^{tmp} , which may contain duplicates and nodes in L_i/L_{i-1} . Cost is $2|L_i| + |N(L_i)|/B$.
2. **Duplicate removal:** Sort L_{i+1}^{tmp} and remove duplicates by scanning. Cost is $sort(|N(L_i)|)$ and $|N(L_i)|/B$ for scanning.
3. **Removing visited nodes:** Finally, we can make L_{i+1} by removing nodes in L_i/L_{i-1} from L_{i+1}^{tmp} . Double concurrent pointers strategy: cost is $O(|N(L_i)|/B + |L_i|/B + |L_{i-1}|/B)$.

Total cost for level L_{i+1} :

$O(|L_i| + |N(L_i)|/B + sort(|N(L_i)|) + |L_i|/B + |L_{i-1}|/B)$.

Total cost:

Note that $\sum_i |L_i| = |V|$ since every node appears in exactly 1 level.

Also, $\sum_i (|N(L_i)|/B) = |E|/B$ and $\sum_i sort(N(L_i)) \leq sort(|E|)$.