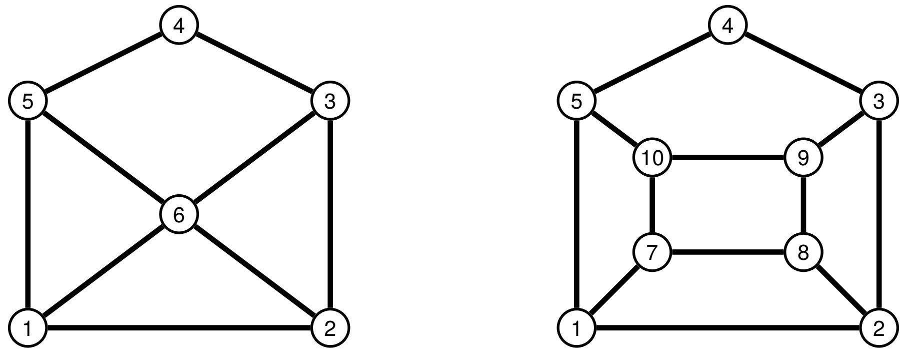
# Take-Home Midterm — *Contemporary Algorithms, T. II/2019–20*

(due: wed feb 26 @ 11:59pm)

**Ground Rules:** There are 3 problems. Do all of them. Solve them by yourself. Typeset your answers and hand in a PDF file electronically to Canvas. You can look things up on the Internet and elsewhere; refrain from blindly copying solutions. Note: If you can't solve a subproblem but need it subsequently, you can pretend you've solved it and invoke it in the following parts. You may earn up to 300 points, but we'll actually grade out of $Q$ points, where $Q$ is to be determined.

**Problem 1. [100 points]** *Unstaring.* We will consider a data structure for simple, undirected graphs that supports the following operations:

- $\textsc{addVertex}()$ adds a new vertex to the graph and returns the vertex identifier;
- $\textsc{addEdge}(u, v)$ adds an edge between vertex $u$ and vertex $v$, if not present; and
- $\textsc{unstar}(u)$ is only defined on vertices with degree at least 3. It replaces vertex $u$ as follows—if $u$ has degree $d = \deg(u)$, then $u$ is replaced with a cycle $C_d$, effectively adding $d$ new vertices and removing $u$. Additionally, each neighbor of $u$ has an edge to a unique vertex of the added cycle. Hence, if $d \geq 3$, each new vertex of $C_d$ ends up with degree 3. For example, calling $\textsc{unstar}(6)$ on the left graph below results in the right graph.



By keeping the graph as an adjacency table (like we did in data structures), each $\textsc{addVertex}$ and $\textsc{addEdge}$ can be supported in $\Theta(1)$ time and each $\textsc{unstar}$ takes $\Theta(d)$ time.

**Your task:** Prove, using the potential method or otherwise, that any sequence of $N$ operations (i.e., any valid combination of $\textsc{addVertex}$, $\textsc{addEdge}$, $\textsc{unstar}$ of length $N$) takes at most $O(N)$ time in total.

**Problem 2. [40+20+40 points]** *Parallel Sublinear-Work Approximate Median.* This problem has two main parts. First, we're going to develop a parallel algorithm to find the median from an array. Then, we're going to look at developing a sampling scheme that will give us an approximate median.

Given an array $A$ of numbers, define the rank of $u$ with respect to $A$, denoted by $\text{rank}_A(u)$, to be the number of elements in $A$ that are less than or equal to $A$. For the purpose of this problem, the *median* of $A$ is the smallest number $x \in A$ such that $\text{rank}(x) \geq \texttt{len}(A)/2$. Simply put, the median is the element $\texttt{sorted}(A)[n//2]$, where $n = \texttt{len}(A)$.

(i) Study the `kthSmallest` algorithm in the handout for parallel algorithms. Let $n = \texttt{len}(A)$. Prove that if randomized pivot selection is used, the median element (i.e., using $k = \lfloor n/2 \rfloor$) can be found in expected $O(n)$ work and $O(\log^2 n)$ span with high probability. (*Hint:* Showing the work bound is the same as what was done in your Data Structures class long time ago; showing the high-probability span bound is similar to the skip-list analysis.)

(ii) Sampling with replacement (see the Internet for more explanation) is when after a sample $x$ is drawn, that $x$ is returned back to the pool and continues to be available for further sampling. This means an item $x$ can be drawn multiple times. Explain how you can sample $T$ elements with replacement from a (random-access) array $A$ of length $n$ in $O(T)$ work and $O(1)$ span, assuming each `randint(low, high)` takes $O(1)$ work and span.

(iii) Let $X$ be an array of $n$ *distinct* numbers. An $\varepsilon$-approximate-median of $X$ is an element $y \in X$ such that $\frac{n}{2} - \varepsilon n < \text{rank}_X(y) < \frac{n}{2} + \varepsilon n$.

**Your Task:** You'll derive an $\varepsilon$-approximate-median that runs in $o(n)$ work (little-O of $n$) and $O(\text{polylog}(n))$ span. Here's an outline of how you want to proceed:

1. For a parameter $T > 0$ (that you'll set), use the previous part to sample $T$ items with replacement from $X$. Call the resulting array $A$.
2. Run the algorithm in part (i) to find the median on $A$.

Because you will use $T = o(n)$, the work required will be $O(T)$. The main line of argument you'll need to show is that for your setting of $T$, good things will happen—i.e., the answer your algorithm returns is an $\varepsilon$-approximate-median with high-probability.

To analyze this, you may find the following setup and observation useful: Fix $\varepsilon > 0$. Let

$$S = \left\{x \in A \mid \text{rank}_X(x) \le \tfrac{n}{2} - \varepsilon n\right\} \qquad L = \left\{x \in A \mid \text{rank}_X(x) \ge \tfrac{n}{2} + \varepsilon n\right\},$$

as determined by their ranks in $X$ (not in $A$). Now convince yourself that if $|S| < T/2$ and $|L| < T/2$, then the median of $A$ is an $\varepsilon$-approximate-median of $X$.

In your write-up, be sure to indicate the value of $T$ you end up with, prove that this setting of $T$ gives you the desired high probability bound, and analyze the total cost (work and span). You don't need to pinpoint the constants in $T$; it's okay to say use $T = \Theta(\cdots)$ or $T = \Omega(\ldots)$.

(*Hint:* Chernoff-Hoeffding and union bounds. Be inspired by P7 on Assignment 1.)

**Problem 3. [100 points]** *Hi, Johnson-Lindenstrauss.* In class, we proved the following lemma:

> **Distributional J-L.** Let $0 < \varepsilon < \frac{1}{2}$. If $A(x) = \frac{1}{\sqrt{k}} Mx$, where each entry of the matrix $M_{k \times D}$ is distributed as $N(0,1)$, and $k = c\varepsilon^{-2}\log(1/\delta)$ for some large enough $c$, then for any $x \in \mathbb{R}^D$ with $\|x\| = 1$,
>
> $$\mathbf{Pr}\left[\|A(x)\|_2^2 \in 1 \pm \varepsilon\right] \ge 1 - \delta.$$

As it turns out, generating $k \times D$ Gaussians requires a lot of random bits and working with real-valued distributions require a lot of care. In this problem, we'll look at replacing $M$ with a much simpler discrete distribution. Instead, $M$ will be a $\{+1, -1\}$-matrix, where each entry of $M$ is $+1$ with probability $\frac{1}{2}$ and $-1$ with probability $\frac{1}{2}$.

**Your Task:** Prove Distributional J-L where the matrix $M$ is replaced with a new, simpler matrix as just described. To begin, you may wish to review your notes and/or your homework 1. In your work, the new bits that you will need to derive/prove are the following:

- For $0 < t < 1/2$, if $R \in_R \{-1, +1\}$, then $\mathbf{E}\left[e^{tR}\right] \le e^{t^2/2}$. As substeps, it helps to note

$$1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \cdots \le \sum_{k \ge 0} \frac{(x^2/2)^k}{k!}.$$

- Let $Y_i$ be the $i$-th row of the vector $A(x)$. Notice that $Y_i$ is simply $Y_i = \frac{1}{\sqrt{k}} V_i$, where

$$V_i = \begin{bmatrix} R_1 & R_2 & \ldots & R_D \end{bmatrix} x = \sum_{j=1}^{D} R_j x_j,$$

where $R_j \in_R \{+1, -1\}$ is drawn independently and uniformly at random. You'll show that

$$\mathbf{E}\left[e^{tV_i}\right] \le \exp\left(\tfrac{1}{2}t^2 \cdot \|x\|_2^2\right) \tag{0.1}$$

- Let $G \sim N(0,1)$. Show that

$$\mathbf{E}\left[e^{tG}\right] = e^{t^2/2}. \tag{0.2}$$

- Finally, you'll prove that for $\|x\|_2 = 1$,

  $$\mathbf{E}\left[e^{tV_i^2}\right] \leq \frac{1}{\sqrt{1-2t}}.$$

  (You may remember that this is the same tail we had for the Gaussian setting.)

  To prove this, there is a super cute (yet super technical) convolution trick that you may find useful. Let's focus on $V = V_i$. Hence, if $f_V(v)$ is the p.d.f. of $V$, then

  $$\begin{aligned}\mathbf{E}\left[tV^2\right] &= \int_{v\in\mathbb{R}} e^{tv^2} f_V(v)dv = \int_{v\in\mathbb{R}} e^{(\sqrt{2t}\cdot v)^2/2} f_V(v)dv \\ &= \int_{v\in\mathbb{R}} \mathbf{E}\left[e^{\sqrt{2t}vG}\right] f_V(v)dv = \mathop{\mathbf{E}}_{V,G}\left[e^{\sqrt{2t}VG}\right] \\ &= \mathop{\mathbf{E}}_{G}\left[\mathop{\mathbf{E}}_{V}\left[e^{(\sqrt{2t}G)V}\right]\right].\end{aligned}$$

  At this point, you can apply equation (0.1), then equation (0.2). Then, you'll use an upper bound of $\mathbf{E}\left[e^{tG^2}\right]$ from class.

Now combine everything together and tie up loose ends.