Many of the sublinear algorithms are approximate and/or randomized. We will see some examples today.

**Diameter of a Metric [Approximate]**

**Input:** \(n\) points and all pairwise distances satisfying triangle inequality.

**Goal:** Compute the diameter of the set, which is the largest pair-wise distance.

**Theorem** (by Indyk): There is a deterministic algorithm that approximates the diameter within factor 2 in time \(O(n)\).

The only requirement is that it’s a metric (so we have the triangle inequality) and the distances is symmetric.

**Algorithm**

Choose 1 point arbitrarily and check the distance between it and all other points. Then take the max.

**Analysis**

**Runtime:** \(O(n)\) - Obvious.

**Correctness:**

Denote \(D_{ij}\) as the distance between point \(i\) and point \(j\).

Suppose \(OPT = D_{ab}\) and suppose the arbitrary point we chose is \(i\).

By the triangle inequality: \(OPT = D_{ab} \leq D_{ai} + D_{ib}\)

At least one of \(D_{ai}\) or \(D_{ib}\) is \(\geq \frac{1}{2} OPT\).

So \(ALG \geq \frac{1}{2} OPT\), which means we have a 2 approximation.

**Finding element in sorted list [Randomized]**

**Input:** Given a list that is sorted but in a linked list structure. However, it also has direct access. (for instance - an array of elements, where each element points at the index of the next element)

**Goal:** Find whether \(q\) appears in the list.

**Theorem** (by Chazelle,-Liu-Magen): There is a randomized algorithm that runs in time \(O(\sqrt{n})\) and is correct with high probability. The error is one sided – so if \(q\) is found it is certainly there. If not, then it is not there with high probability.
Note: With high probability we mean that it's bigger than \( \frac{2}{3} \). One can later amplify it if needed.

**Algorithm**

Define \( t = 2\sqrt{n} \)

1. Scan the first \( t \) elements of the list. If \( q \) was found report it was found.
2. Choose at random \( k = \sqrt{n} \) elements from the list
3. Find which of them is \( \leq q \) and take the largest
4. Scan the linked list starting from this element for the next \( t \) elements and report whether \( q \) was found or not.

**Analysis**

**Runtime:** Obviously \( O(k + t) = O(\sqrt{n}) \)

**Correctness:** wlog, \( q \) in the list. Since if not we will certainly not find it and return the right answer. Let the linked list be: \( a_1 < a_2 < \ldots < a_n \) and suppose that \( q = a_j \)

\[
\Pr\left[ \text{none of the } k \text{ samples } \in \left\{ a_{j-t+1}, \ldots, a_j \right\} \right] \leq \left( 1 - \frac{1}{n} \right)^k \leq e^{-\frac{tk}{n}} \leq \frac{1}{7}.
\]

It follows that with probability over \( \frac{6}{7} \) the algorithm will sample at least one of \( q_{j-t+1}, \ldots, a_j = q \) in which case the scan will find \( q \).

We can even refine the argument. For instance, we can have a witness for not having \( q \) in the list if when scanning we go from a value smaller than \( q \) to a value that is larger. In addition, we can say we scan the list until we find \( q \) (or find it’s not there) and thus the algorithm will always return the right answer but the runtime is randomized (with a small expectation).

**Approximate average degree in a graph**

**Input:** A connected graph given as an adjacency list.

**Goal:** Compute the average degree in the graph.

**Theorem** [A weaker version of a theorem by Feige]: There is a randomized algorithm that approximates the average degree within a factor of \( 2 + \varepsilon \) (for any desired \( \frac{1}{2} > \varepsilon > 0 \)) in time \( O\left( \left( \frac{1}{\varepsilon} \right)^{O(1)} \cdot \sqrt{n} \right) \)

**Algorithm**

1. Choose a set \( S \) by picking at random \( S = \left( \frac{1}{\varepsilon} \right)^{O(1)} \cdot \sqrt{n} \) vertices.
2. Compute the average degree \( -d_s \)
3. Repeat the above $\frac{8}{\epsilon}$ times and report the smallest value in step 2.

**Analysis**

**Runtime:** $O\left(\frac{1}{\epsilon} O(1) \cdot \sqrt{n}\right)$ — obvious.

**Correctness:** Let $d_s$ be the average degree of $S$, and let $d$ be the average degree in $G$

**Lemma 1:** In one iteration:

$$\Pr[d_s < \frac{1}{2} (1 - \epsilon) d] \leq \frac{\epsilon}{64}$$

**Lemma 2:** In one iteration:

$$\Pr[d_s > (1 + \epsilon) d] \leq 1 - \frac{\epsilon}{2}$$

Given these two lemmas this is how you prove the theorem:

$$\Pr[ALG > (1 + \epsilon) d] \leq \left(1 - \frac{\epsilon}{2}\right)^{\frac{8}{\epsilon}} < e^{-4} < \frac{1}{8}$$

$$\Pr\left[ALG < \frac{1}{2} (1 - \epsilon) d\right] \leq \frac{8 \cdot \epsilon}{64} = \frac{1}{8} \Rightarrow$$

Algorithm achieves approximation $2 + \epsilon$ with probability $\geq \frac{3}{4}$.

**Proof of lemma 2:**

Denote $s = |S|$

Let $X_i$ for $i = 1, ..., s$ be the degree of the $i$'th vertex chosen to $S \Rightarrow d_s = \frac{1}{s} \sum_{i=1}^{s} X_i$ and so:

$$E[d_s] = \frac{1}{s} \sum_{i=1}^{s} E[X_i] = d$$

**Markov's inequality:**

If $Z \geq 0$ is a random variable, then for all $\alpha > 1$:

$$\Pr[Z \geq \alpha E[Z]] \leq \frac{1}{\alpha}$$

So by using Markov's inequality we get:

$$\Pr[d_s \geq (1 + \epsilon) d] \leq \frac{1}{1 + \epsilon} < 1 - \frac{\epsilon}{2}$$

**Proof of lemma 1:**

Let $H$ be the set of $\sqrt{\epsilon n}$ vertices with the highest degree.
Let \( L = V \setminus H \).

Wlog, we assume \( S \) is chosen from \( L \) (the true \( d_s \) dominates this analysis).

So now, let \( X_i \) for \( i = 1, \ldots, s \) be the degree of \( i \)th vertex chosen.

\[
d_s = \frac{1}{s} \sum_{i=1}^{s} X_i
\]

**Chernoff bound:**

Let \( Z_i \in \{0,1\} \) for \( i = 1, \ldots, s \) be independent random variables. Then for all \( 0 < \delta < 1 \):

\[
\Pr \left[ \sum_{i=1}^{s} Z_i \leq (1 - \delta) \cdot E \left[ \sum_{i=1}^{s} Z_i \right] \right] \leq e^{-\delta^2 \frac{E[\sum Z_i]}{4}}
\]

Denote \( d_H \) to be the smallest degree in \( H \).

Then \( 1 \leq X_i \leq d_H \)

\[
\Pr \left[ d_s \leq (1 - \epsilon) E[d_s] \right] = \Pr \left[ \frac{\sum X_i}{d_H} \leq (1 - \epsilon) E \left[ \frac{\sum X_i}{d_H} \right] \right] \leq e^{-\epsilon^2 \frac{E[\sum Z_i]}{4}} = e^{-\epsilon^2 \frac{E[\sum X_i]}{4d_H}}
\]

\[
E \left[ \sum X_i \right] = |S| \cdot \frac{E[X_1]}{average degree in L}
\]

So now we would like to find the size of \( S \) such that we’ll reach our bound. Thus, we’ll split into cases based on \( d_H \)

**Case 1 -** \( d_H \geq \frac{1}{\epsilon} |H| \):

Note the following facts:

(*) Each vertex in \( |H| \) has a degree that is higher than \( d_H \) so the sum of all the degrees of vertices in \( |H| \) is larger than \( |H| \cdot d_H \)

(**) The maximal number of edges of \( H \) that have both their ends in \( H \) is the number of possible pairs of vertices of \( H \cdot \binom{|H|}{2} \), and so the contribution of those edges to the degrees of the vertices of \( H \) is at most \( 2 \cdot \binom{|H|}{2} = |H|(|H| - 1) \leq |H|^2 \)

\[
E[X_1] \geq \frac{d_H |H| - |H|^2}{|L|} = \frac{(d_H - |H|) \cdot |H|}{|L|} = \left( 1 - \frac{|H|}{d_H} \right) \cdot d_H \cdot |H| \geq \frac{n > |L|}{\epsilon} (1 - \epsilon) \cdot d_H \cdot |H| / n
\]

So in this case:
\[ e^{-\epsilon^2 \cdot \frac{E[\sum X_i]}{4 \cdot d_H}} \leq e^{-\epsilon^2 \cdot \frac{s(1-\epsilon) \cdot \#H}{4 \cdot d_H}} \]

Enough to have (up to constants and \( \log \frac{1}{\epsilon} \) factors):

\[ \frac{s \cdot \epsilon^2 \cdot |H|}{n} \geq 1 \]

To get our desired bound.

This implies that it satisfies to have:

\[ s \geq \epsilon^{-2} \cdot \frac{n}{|H|} \cdot \frac{|H|=\sqrt{n}}{\sqrt{n}} \cdot (1 - \epsilon)^{O(1)} \cdot \sqrt{n} \]

To be continued next class...