1 Approximate average degree in a graph - Cont.

Last time we’ve seen an algorithm for estimating the average degree in a graph.

**Theorem 1 (Feige)** There is a randomized algorithm that approximates the average degree within a factor of $2 + \epsilon$ for an $\epsilon \in (0, \frac{1}{2})$ in time $\left(\frac{1}{\epsilon}\right)^{O(1)} \cdot \sqrt{n}$.

The algorithm is as following:

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

The only thing left to show from last week is the following claim.

**Claim 2** Let $d_S$ be the average degree of $S$, and let $d$ be the average degree in $G$. Then

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

**Proof** Let $H \subseteq V$ be the $\sqrt{n}$ vertices of the highest degree and let $L = V \setminus H$. We assume that $S$ is sampled from $L$ as this distribution is dominated by the actual $d_S$. Then

$$\mathbb{E}[d_S] \geq \frac{1}{2} \cdot \frac{d|V| - |H|^2}{|L|} \geq \frac{1}{2} \cdot \frac{(d - \epsilon)|V|}{|V|} = \frac{1}{2}(d - \epsilon)$$

where $\frac{1}{2}$ comes from counting the number of edges touching $L$ with at least one endpoint. Let $d_H$ be the lowest degree in $H$ and let $1 \leq X_i \leq d_H$ be the degree of the $i$'th sampled vertex. Then $d_S = \frac{1}{s} \sum X_i$ and by Chernoff (multiplicative) bound we have

$$\Pr[d_S < \frac{1}{2}(1 - \epsilon)d] = \Pr[\frac{1}{s} \sum X_i < (1 - \epsilon)\mathbb{E}[\frac{1}{s} \sum X_i]] \leq \exp(-\frac{\epsilon^2 \cdot s \cdot \mathbb{E}[X_1]}{4d_H})$$

By taking $s \geq \epsilon^{-2} \frac{d_H}{\mathbb{E}[X_1]}$, we would get the required result. But we need $s$ to be independent of $d_H$ and $\mathbb{E}[X_1]$. We analyze two cases below.
Case 1 \( d_H \geq \frac{1}{\epsilon} |H| \). Then

\[
\mathbb{E}[X_1] = \sum_{v \in L} d(v) \geq |H|d_H - |H|^2 \geq |H|(d_H - |H|) \geq |H|(1 - \epsilon)d_H
\]

Implies \( \frac{d_H}{\mathbb{E}[X_1]} \leq \frac{|V|}{|H|} \) and thus

\[
\epsilon^{-2} \frac{d_H}{\mathbb{E}[X_1]} \leq \epsilon^{-2} \frac{|V|}{|H|} = \text{poly}(\frac{1}{\epsilon})\sqrt{n} = s
\]

Case 2 \( d_H < \frac{1}{\epsilon} |H| \). Since \( \mathbb{E}[X_1] \geq 1 \) we may take

\[
\epsilon^{-2} \frac{d_H}{\mathbb{E}[X_1]} \leq \epsilon^{-3} \sqrt{\epsilon n} = \text{poly}(\frac{1}{\epsilon})\sqrt{n} = s
\]

So by taking \( s \) to be the largest of the above two, we complete the proof of the theorem. \( \blacksquare \)

Remark In fact there is a matching lower bound under the assumption that the algorithm is allowed to observe only the degrees of a vertex. Any algorithm that uses only degree queries and estimates the average degree within a ratio \( 2 - \delta \) for some constant \( \delta \) requires \( \Omega(n) \) queries.

2 Minimum spanning trees

Given an undirected connected graph \( G \) with maximal degree \( \leq D \) and edge weights in \( \{1, \ldots, W\} \), represented as an adjacency list, we want to compute the weight of \( \text{MST}(G) \).

Theorem 3 (Chazelle, Rubinfeld, Trevisan) There is an algorithm that approximates the cost of \( \text{MST}(G) \) within factor of \( 1 + \epsilon \) in time \( O\left(\left(\frac{1}{\epsilon}\right)W^3 D \lg(n)\right)\).

We start proving the theorem with the following lemma.

Lemma 4 Let \( G_i \) be the subgraph of \( G \) containing all edges of weight at most \( i \). Let \( c_i \) be the number of connected components in \( G_i \). Then

\[
\text{MST}(G) = n - W + \sum_{i=1}^{W-1} c_i
\]

Next we present an algorithms for approximating \( \text{MST}(G) \).

1. For \( i = 1, \ldots, W \) estimate \( \hat{c}_i \) for \( c_i \) within additive error of \( \frac{\epsilon n}{W} \)

2. Report \( n - W + \sum_{i=1}^{W-1} \hat{c}_i \)

Observe that assuming that step 1 succeeds w.p. \( \geq 1 - \frac{1}{4W} \) for all \( i \), the algorithm estimates \( \text{MST}(G) \) within factor of \( (1 + \epsilon) \) with probability at least \( \frac{3}{4} \).
2.1 Estimating $c_j$

We may assume that the input is the graph $G_j$ by ignoring the edges of weight $w_e \geq j$. The algorithms for estimating $c_j$ is as following:

1. Choose $s = \epsilon^{-2}W^3$ random vertices $v_1, \ldots, v_s$.

2. For each $v_i$ do
   • Choose r.v. $1 \leq X_i \leq n$ such that $\Pr[X \geq k] = \frac{1}{k}$ for all $k = 1, \ldots, n$
   • Perform BFS from $v_i$ until either the entire connected component is explored and set $b_i = 1$, or until explored $X_i + 1$ distinct vertices and set $b_i = 0$.

3. Report $\hat{c} = \frac{n}{c} \sum_{i=1}^{s} b_i$

The runtime is clearly $s \cdot D \cdot \mathbb{E}[X]$, where $D$ comes from the BFS, and $\mathbb{E}[X] = O(\log(n))$.

We first compute $\mathbb{E}[\hat{c}]$. Let $c$ be the number of connected components. Then

$$\mathbb{E}[\hat{c}] = \frac{n}{s} \sum_{i=1}^{s} \mathbb{E}[b_i] = n\mathbb{E}[b_1]$$

So it is enough to compute $\mathbb{E}[b_1]$:

$$\mathbb{E}[b_1] = \sum_{C} \frac{|C|}{n} \Pr[X \geq |C|] = \sum_{C} \frac{|C|}{n} \cdot \frac{1}{|C|} = \frac{c}{n}$$

where the sum runs over all connected components $C$. Therefore

$$\mathbb{E}[\hat{c}] = n\mathbb{E}[b_1] = c$$

Next we show that the variance of $\hat{c}$ is not too large, and use Chebyshev inequality to conclude that $\hat{c}$ estimates $c$ well.

$$\text{Var}[b_1] = p(1-p) = \frac{c}{n} \left(1 - \frac{c}{n}\right) \leq \frac{c}{n}$$

$$\text{Var}[\hat{c}] = \frac{n^2}{s^2} \sum_{i=1}^{s} \text{Var}[b_i] \leq \frac{n^2}{s^2} \sum_{i=1}^{s} \frac{c}{n} = \frac{nc}{s}$$

By Chebyshev inequality we have

$$\Pr[|\hat{c} - c| \geq \frac{en}{W}] \leq \frac{\text{Var}[\hat{c}]W^2}{n^2\epsilon^2} \leq \frac{cW^2}{s\epsilon^2n} \leq \frac{1}{4W}$$

3 Maximum matching

In the problem of maximum matching we are given an undirected connected graph $G$ with maximal degree $\leq D$. The goal is to find a maximum matching, that is a maximum set of disjoint edges.
Lemma 5  The size of any maximal matching is at least half of a maximum matching.

Theorem 6 (Nguyen, Onak) There is an algorithm that $(1/2, \epsilon)$-approximates maximal matching in time $D^{O(D)}/\epsilon$. That is with high probability the algorithm returns a value in the range $[\frac{1}{2}M - \epsilon n, M]$, where $M$ is the size of maximum matching.

Let us first consider the following greedy algorithm for finding maximal matching:

1. Start with an empty set $M = \emptyset$.
2. Iterate over the edges in an arbitrary order and add an edge to $M$ if possible.
3. Output $M$.

Our sublinear algorithm will simulate this greedy algorithm. We will fix some order of the edges and by sampling edges we will add an edge to $M$ if all its neighbors are not in $M$. The algorithm is the following:

1. For each edge $e$ choose a permutation of edges by assigning each edge a random priority $p(e) \sim U[0, 1]$ uniformly.
2. Choose $s = O(D\epsilon^{-2})$ edges $e_1, \ldots, e_s$ uniformly at random from $V \times \{1, \ldots, D\}$.
3. For each $i = 1, \ldots, s$ explore the neighbors of $e_i$ inductively. Set $X_i = 1$ if none of its neighbors is in the matching and $X_i = 0$ otherwise.
4. Output $\sum \frac{X_i}{s} \cdot Dn$. 

2-4