Lecture 5:

Estimating the Number of Connected Components

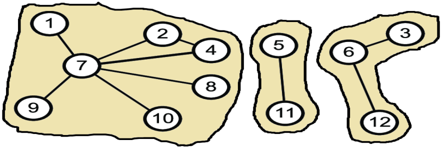
Course: Algorithms for Big Data

Instructor: Hossein Jowhari

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Spring 2021

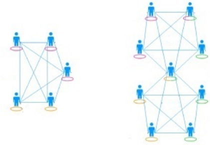
Connected Components

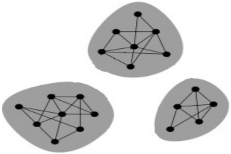


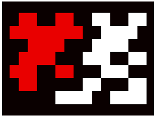
- A pair of vertices u and v are connected iff there is a path between u and v.
- In an undirected graph $G = (V, E)$, a connected component in G is a maximal subset $S \subseteq V$ where every pair of vertices in S are connected in the induced subgraph on S.
- The above graph has 3 connected components.

Connected components can represent:

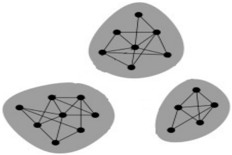
- Related entities in a system
- Communities in a social network
- Clusters in a graph
- Objects in an image







Computing the connected components



- A relatively easy problem
- There is a $O(m)$ time algorithm using popular graph traversal algorithms (BFS/DFS)
- $cc(G)$ = number of connected components in graph G
- Every algorithm that distinguishes between $cc(G) = 1$ and $cc(G) \geq 2$ needs $\Omega(m)$ time.
- An additive approximation of $cc(G)$ is possible in sublinear time.

Additive approximation of $cc(G)$

We study a randomized algorithm A that approximates the number of connected components in the graph G.

Let $A(G)$ be the output of the algorithm. We have

$$Pr\Big(|A(G) - cc(G)| \le \epsilon n\Big) \ge 1 - \delta$$

Algorithm A performs $O(\ln(\frac{1}{\delta})\frac{1}{\epsilon^4})$ number of neighbor queries.

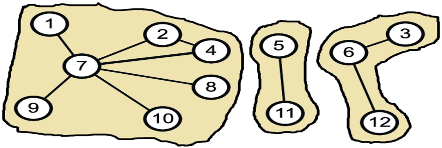
Neighbor query: Given a vertex u and number $i \in \{1, \ldots, n\}$, output the i-th neighbor of u if it exists otherwise output $\perp$.

$$u : \underbrace{(v_1, \ldots, v_k)}_{\text{neighbors of } u}$$

Idea Behind Algorithm

Let $C(u)$ be the connected component that contains the vertex u. Here $|C|$ denotes the size of component C.

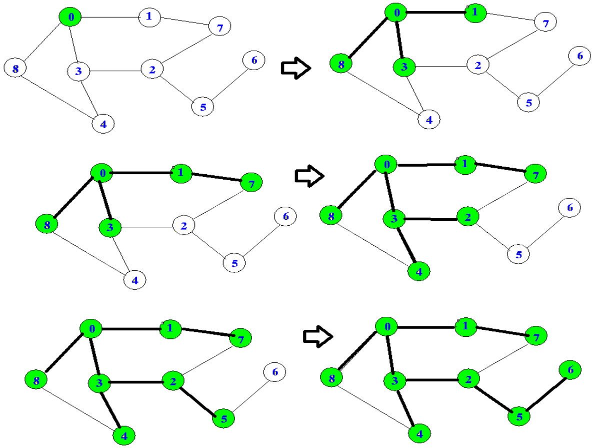
Lemma 1: $cc(G) = \sum_{u \in V} \frac{1}{|C(u)|}$.



In the above example:

$$cc(G) = (\underbrace{\frac{1}{7} + \ldots + \frac{1}{7}}_{7}) + (\frac{1}{2} + \frac{1}{2}) + (\frac{1}{3} + \frac{1}{3} + \frac{1}{3}) = 3$$

A slow algorithm: For each vertex $u \in V$, compute $\frac{1}{|C(u)|}$ and output the resulting summation.



BFS: Starting from vertex 0, we find all vetices that are reachable.

We check every reachable edge.

Time complexity for each vertex is $O(m)$

Time complexity of the slow algorithm: $O(nm)$

Additive Approximation of $cc(G)$

A deterministic algorithm with additive error: For each vertex $u \in V$, compute $\frac{1}{|C(u)|}$ but stop when the size of the component exceeds $\frac{2}{\epsilon}$. In other words, we compute the following.

$$cc'(G) = \sum_{u \in V} \frac{1}{z(u)}, \qquad \text{where } z(u) = \min\{|C(u)|, \frac{2}{\epsilon}\}.$$

Claim:

$$|cc'(GG) - cc(G)| \leq \frac{\epsilon n}{2}.$$

Proof of Claim: The quantity $cc'(G)$ overestimates $cc(G)$.

So we have

$$cc'(G) - cc(G) =$$

$$= \sum_{u \in V} \frac{1}{\min\{|C(u)|, \frac{2}{\epsilon}\}} - \sum_{u \in V} \frac{1}{|C(u)|}$$

$$= \Big(\sum_{|C(u)| \leq \frac{2}{\epsilon}} \frac{1}{|C(u)|} + \sum_{|C(u)| > \frac{2}{\epsilon}} \frac{\epsilon}{2} \Big) - \Big(\sum_{|C(u)| \leq \frac{2}{\epsilon}} \frac{1}{|C(u)|} + \sum_{|C(u)| > \frac{2}{\epsilon}} \frac{1}{|C(u)|} \Big)$$

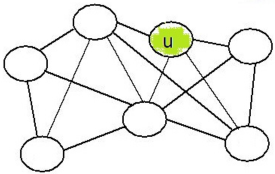
$$= \sum_{|C(u)| > \frac{2}{\epsilon}} \Big(\frac{\epsilon}{2} - \frac{1}{|C(u)|} \Big)$$

$$< \sum_{|C(u)| > \frac{2}{\epsilon}} \frac{\epsilon}{2}$$

$$\leq \frac{\epsilon n}{2}$$

The running time of the deterministic algorithm: For each vertex u, we compute $\min\{|C(u)|, \frac{2}{\epsilon}\}$.

Question: In worst-case, how many neighbor queries we need ask to compute $\min\{|C(u)|, \frac{2}{\epsilon}\}$?



A graph with k vertices has at most $\frac{1}{2}k(k-1)$ edges.
Therefore we need to ask at most $\frac{2}{\epsilon} \times \frac{2}{\epsilon} + 1$ neighbor queries.

Query complexity of the deterministic algorithm: $O(\frac{n}{\epsilon^2})$

A faster randomized algorithm

Consider the quantity,

$$cc'(G) = \sum_{u \in V} \frac{1}{z(u)}, \qquad \text{where } z(u) = \min\{|C(u)|, \frac{2}{\epsilon}\}.$$

Let $z'(u) = \frac{1}{z(u)}$. We have

$$cc'(G) = \sum_{u \in V} z'(u), \qquad \text{Note: } z'(u) \in [\frac{\epsilon}{2}, 1].$$

Randomized algorithm: Sample $S \subseteq V$ (with replacement), and compute

$$\frac{n}{|S|} \sum_{u \in S} z'(u)$$

Analysis of the randomized algorithm:

Let $i \in \{1, \ldots, |S|\}$. Let X_i be the outcome of the i-th sample. We have

$$E[X_i] = \frac{1}{n}(z'(u_1) + z'(u_2) + \ldots + z'(u_n)) = \frac{1}{n}cc'(G).$$

Let $X = \sum_{i=1}^{|S|} X_i$. We have

$$E[X] = \sum_{i=1}^{|S|} E[X_i] = \frac{|S|}{n}cc'(G)$$

Let Y be the output of the algorithm. We have $Y = \frac{n}{|S|}X$. Therefore,

$$E[Y] = \frac{n}{|S|}E[X] = cc'(G).$$

$$Pr\Big(|Y - E[Y]| \geq \frac{\epsilon n}{2}\Big) = Pr\Big(|\frac{nX}{|S|} - E[\frac{nX}{|S|}]| \geq \frac{\epsilon n}{2}\Big)$$

$$= Pr\Big(|X - E[X]| \geq \frac{\epsilon |S|}{2}\Big)$$

Additive Chernoff Bound: Suppose $Y_1, \ldots, Y_k$ are independent random variables taking values in the interval $[0,1]$. Let $Y = \sum_{i=1}^{k} Y_i$. For any $t \geq 1$,

$$Pr\Big(|Y - E[Y]| \geq t\Big) \leq 2e^{-\frac{2t^2}{k}}$$

Using additive Chernoff bound, we get

$$Pr\Big(|X - E[X]| \geq \frac{\epsilon |S|}{2}\Big) \leq 2e^{-\frac{2(\frac{\epsilon^2 |S|^2}{4})}{|S|}} \leq \delta \;\; \Rightarrow \;\; s = \Omega\Big(\frac{1}{\epsilon^2}\ln(\frac{1}{\delta})\Big)$$

Y is the output of the algorithm. We just showed that

$$Pr\Big(|Y - E[Y]| \geq \frac{\epsilon n}{2}\Big) \leq \delta$$

Since $E[Y] = cc'(G)$,

$$Pr\Big(|Y - cc'(G)| \geq \frac{\epsilon n}{2}\Big) \leq \delta$$

Also we have,

$$cc'(G) - cc(G) \leq \frac{\epsilon n}{2}$$

Finally,

$$Pr\Big(|Y - cc(G)| \geq \epsilon n\Big) \leq \delta$$

Query complexity of the algorithm:

- We sample $s = \Omega(\frac{1}{\epsilon^2}\ln(\frac{1}{\delta}))$ vertices.
- For each sampled vertex u, we compute $z'(u)$.
- As we saw earlier, to compute $z'(u)$, we make at most $O(\frac{1}{\epsilon^2})$ neighbor queries.
- The total number of neighbor queries is $O(\ln(\frac{1}{\delta})\frac{1}{\epsilon^4})$.

Final Remarks

- The algorithm we presented is from the following paper:

  B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. SIAM J. of Computing. 2005.

- Estimating the number of connected components is used in estimating the weight of the minimum spanning tree of a graph.

- There are faster algorithms for estimating the number of connected components. See the following paper.

  P. Berenbrink, B. Krayenhoff, F. Mallmann-Trenn. Estimating the number of connected components in sublinear time. Information Processing Letters. 2014.