Lecture 11

TTIC 41000: Algorithms for Massive Data

Toyota Technological Institute at Chicago

Spring 2021

Instructor: Sepideh Mahabadi

This Lecture

- ❑ Sublinear Time Model of Computation
- ❑ Distinct Elements
- ❑ Graph Connectivity
- ❑ Approximating the average degree of a graph

Sublinear Time Algorithms

- The input is so huge that even reading all of it is not feasible
- Solve the problem accessing a *small* portion of the input
  - Need to specify the access model: what queries can be asked?
    - Random Access
      - E.g., For an array, given i, return the ith entry of a matrix, i.e., A[i]
      - For a graph, query the adjacency graph: given u,v, return A[u][v], i.e., does there exist an edge between u and v
      - Adjacency List: given u, i, return the ith neighbor of the vertex u (or Null if deg(u)<i)
    - Sample
      - Algorithm receives a random sample from a specific distribution
  - Parameters of interest
    - Number of queries asked
    - Actual runtime (could be sublinear, polynomial, or even exponential)

Example Goals

- Estimate the solution to a problem
  - E.g. what is the average degree in the graph
  - E.g. what is the size of the minimum set cover
- Property Testing: Testing whether the input has a property P, or is far from having the property
  - does the input need to change a lot to have the property
  - total variation distance between a distribution and the closest distribution having the property

Distinct Elements

Property Testing Algorithm for Distinct Elements

Input: n elements

Output: distinguish

- Yes: all elements are distinct
- No: number of distinct elements $\leq (1-\epsilon)n$
- Otherwise report anything

Algorithm?

- Sample a few elements
- If there are any duplicates say FAIL, otherwise say PASS

Analysis

- Always outputs correctly in the Yes case
- How many samples to succeed in the No case? $O(\sqrt{n}/\epsilon)$

Claim: $O(\sqrt{n}/\epsilon)$ suffices

Intuitively

- ❑ 1,1,2,2,3,3,4,4,…,n/2,n/2
  - How many samples are needed to detect it is a no case?
    - By birthday paradox $O(\sqrt{n})$
      - Birthday paradox: if we take $c\sqrt{n}$ uniformly random samples from [n], with a constant probability, we draw one number twice
- ❑ 1,1,2,2,3,3,ϵn, ϵn, ϵn+1, ϵn+2,…,n- ϵn
  - For the same reason, $O(\sqrt{n}/\epsilon)$
- ❑ 1,1,1,1,..,1,2,3,…, n-ϵn
  - Change it to the previous case: $\boxed{1_1, 1_1}, \boxed{1_2, 1_2}, \dots, \boxed{1_{\frac{\epsilon n}{2}}, 1_{\frac{\epsilon n}{2}}}, 2,3,4, \dots, n - \epsilon n$
- ❑ $\boxed{1,1}$,1,$\boxed{2,2}$,$\boxed{3,3}$,$\boxed{3,3}$,4,5,$\boxed{6,6}$,$\boxed{7,7}$,7,… -> $\boxed{1,1}$,$\boxed{2,2}$,$\boxed{3_1, 3_1}$, $\boxed{3_2, 3_2}$, 4,5,$\boxed{6,6}$,$\boxed{7,7}$,…
- ❑ Assume: there are $\epsilon n/4$ pairs of duplicates

Formal Proof

- $\boldsymbol{S_1}$: first sample $O(\sqrt{n})$ elements
- $\boldsymbol{S_2}$: next sample $O(\sqrt{n}/\epsilon)$ elements

Overall structure of the proof:

- Pair repeated elements (ignoring one in each odd number of repetitions).
- Claim 1: S_1 hits many (i.e., $O(\epsilon\sqrt{n})$) elements which are the first element of a duplicate pair
- Claim 2: S_2 will hit second element of one of those pairs with a constant probability

Proof of Claim 1

- S_1: first $O(\sqrt{n})$ samples,
- Claim 1: S_1 hits many (i.e., $O(\epsilon\sqrt{n})$) elements which are the first element of duplicate pairs
    - Number of pairs: $\epsilon n/4$
    - Probability of hitting the first element in the pair in a random draw of S_1 is $O(\frac{\epsilon}{4})$
    - Expected number of such hits is $(\frac{\sqrt{n}\epsilon}{4})$
    - Concentration: Since the draws are independent, using Chernoff, the number of such hits is at least $(\frac{\sqrt{n}\epsilon}{8})$ w.h.p.
    - What about repetitions?
        - For any two draws, it happens with probability at most $O(\frac{1}{n})$
        - The expected number of total repetitions is at most $\left(\frac{O(\sqrt{n})^2}{n}\right) = O(1)$
        - By Markov, w.p. at least $1 - 1/10$ the total number of repetitions does not increase $O(1)$
    - W.p. $1 - 1/5$, S_1 hits $\Omega(\sqrt{n}\epsilon)$ pairs

Proof of Claim 2

- S_1: first $O(\sqrt{n})$ samples,
- S_2: next $O(\sqrt{n}/\epsilon)$ samples,
- Claim 1: S_1 hits many (i.e., $O(\epsilon\sqrt{n})$) elements which are the first element of duplicate pairs
- Claim 2: S_2 will hit second element of one of those pairs with a constant probability
  - The probability of not hitting the second element in the pairs hit by S_1 is at most

$$\left(1-\frac{\epsilon\sqrt{n}}{cn}\right)^{c_2\sqrt{n}/\epsilon} \leq \left(1-\frac{\epsilon}{c\sqrt{n}}\right)^{\frac{c_2\sqrt{n}}{\epsilon}} \leq e^{-c_2/c}$$

  - Set c_2 large enough such that the above prob. is at most 1/5
- The overall success probability is 1-1/5-1/5=3/5
- As always can boost the success probability by repeating the whole algorithm multiple times.

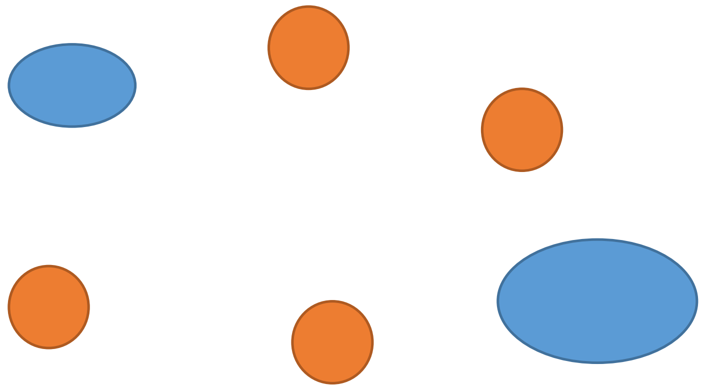
Graph connectivity

Problem

- **Input:** a graph G: n vertices, max degree d,
- **Output**: decide if
  - G is connected
  - $\epsilon -$ far from being connected, i.e., need to add at least ϵdn edges to make it connected
- **Query Model:** adjacency list query model
  - Degree query: given v, what is $\deg(v)$
  - Neighbor query: given v, i, what is ith neighbor of v
- Randomized:
  - Output PASS if it is connected
  - Output FAIL w.p. $\geq 3/4$ if it is far from being connected

Intuitively

- If the graph is far from being connected
  - There are many connected components
  - Many of them should have small size
  - Many nodes in small components



- What can we do?
  - Randomly sample vertices and check the sizes of their connected components

Algorithm

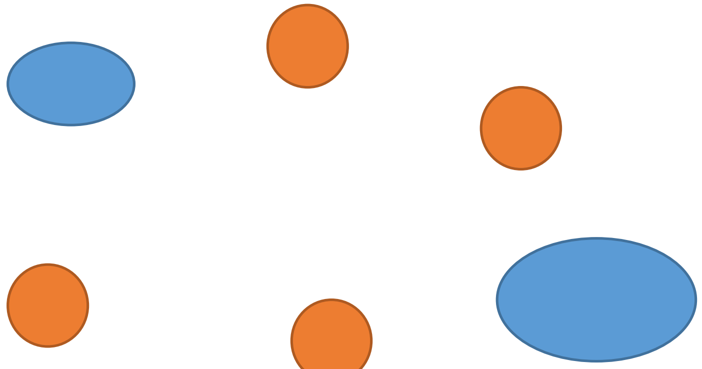
- For $O(\frac{1}{\epsilon d})$ iterations
  - Pick a random node s and run BFS from it until either
    - $\geq (\frac{2}{\epsilon d})$ distinct nodes are encountered
    - Or find that the size of the connected component is less than $\frac{2}{\epsilon d}$
      - Output FAIL
- Output PASS

➢ Runtime: $O\left(\frac{1}{\epsilon d} \cdot \frac{2}{\epsilon d} \cdot d\right) = O(\frac{1}{\epsilon^2 d})$

➢ Correctness:

- Clearly if it is connected, it passes the test
- What happens if it is ϵ −far from being connected?

Analysis



- If the graph is ϵ −far from being connected
  - At least ϵdn edges should be added to make the graph connected
  - There are at least ϵdn connected components in the graph
  - The number of connected components of size larger than $\frac{2}{\epsilon d}$ is at most $\epsilon dn/2$
    - Otherwise the total number of vertices would be larger than $\frac{2}{\epsilon d} \cdot \frac{\epsilon dn}{2} = n$
  - Each such component has at least one vertex in it
  - The total number of vertices in components of size at most $\frac{2}{\epsilon d}$, is at least $\frac{\epsilon dn}{2}$
  - $\Pr[\text{PASS}] \leq \left(1 - \frac{\epsilon dn}{2n}\right)^{\frac{c}{\epsilon d}} \leq e^{-\frac{c}{2}} \leq 1/4$ for large enough c

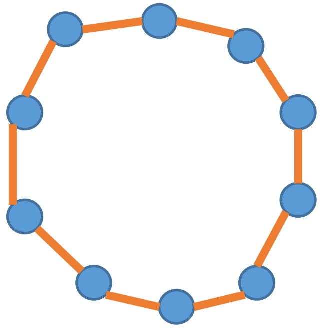
➢ Algorithm succeeds w.p. $\geq 3/4$

Average degree computation

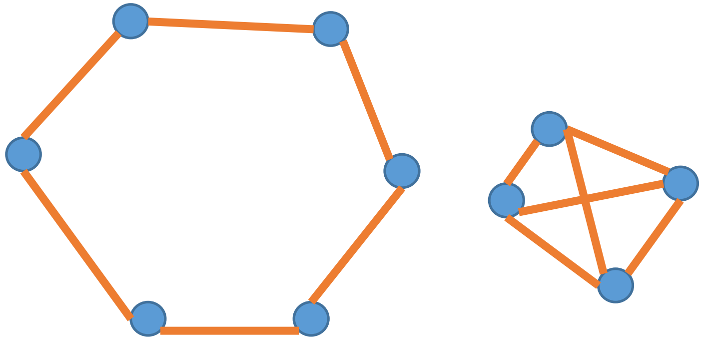
Average degree problem

- **Input:** a graph
- **Output:** the average degree over all vertices $\bar{d}$
- **Query Model:**
  - given v, what is $\deg(v)$
  - given v, i, what is ith neighbor of v
- Naïve algorithm?
  - Sample a few vertices, and report their average degree
  - Problem?
    - Degrees are in the range [n], and can have high variance. E.g. the star graph
    - This requires $\Omega(n)$ samples
  - Solution?
    - In general No, e.g. detecting an empty graph (avg deg =0) vs a graph with a single edge (avg deg=1/n) requires $\Omega(n)$ samples
    - Group vertices of similar degrees into buckets (works assuming graph has $\Omega(n)$ edges).

Lower bound of $\Omega(\sqrt{n})$



- G: a cycle of length n
  - Avg deg is 2
- G': union of i) a cycle of length $n - c\sqrt{n}$, and ii) a clique of size $c\sqrt{n}$
  - Avg deg is $\frac{2(n-c\sqrt{n})+(c\sqrt{n})^2-c\sqrt{n}}{n} \geq 2 + c - \frac{3c}{\sqrt{n}} \geq 1 + c$
- Need $\Omega(\sqrt{n})$ queries to find a clique node



Algorithm

- $\beta = \epsilon/c$, bucket the vertices based on powers of $(1+\beta)$
  - $\boldsymbol{B_i} = \{v: (1+\beta)^{i-1} \leq \deg(v) \leq (1+\beta)^i\}$
  - Number of buckets is $\boldsymbol{t} = \frac{\log n}{\beta} \leq O\left(\frac{\log n}{\epsilon}\right)$
- Total degree
  - Total degree in bucket i is $(1+\beta)^{i-1}|B_i| \leq \deg_{B_i} \leq (1+\beta)^i|B_i|$
  - Overall total degree is $\sum_i (1+\beta)^{i-1}|B_i| \leq deg_{total} \leq \sum_i (1+\beta)^i |B_i|$
- **Algorithm**
  - Sample a set of vertices $\boldsymbol{S}$
  - Let $\boldsymbol{S_i}$ be the samples from the ith bucket
  - Estimate average degree of B_i using S_i
    - $\boldsymbol{\rho_i} = \frac{|S_i|}{|S|}$, then $\mathbb{E}[\rho_i] = \frac{|B_i|}{n}$. Thus **report** $\sum_i \rho_i (1+\beta)^{i-1}$
    - Problem: for i s.t. $|S_i|$ is small, the estimate is off
    - Solution? Ignore vertices in such buckets!

Algorithm

Algorithm

- Sample a set of vertices S
- Let $S_i = B_i \cap S$ be the samples from the ith bucket
- If $|S_i| \geq \sqrt{\frac{\epsilon}{n}} \cdot \frac{|S|}{c \cdot t}$ (this means $|S| > \sqrt{n/\epsilon} \cdot t$)
  - $\rho_i = \frac{|S_i|}{|S|}$
- Else $\rho_i = 0$
- **Output** $\sum_i \rho_i (1+\beta)^{i-1}$

➢ Not over estimating:

  ➢ if $\rho_i = \frac{|B_i|}{n}$, then $\sum_i \rho_i (1+\beta)^{i-1} = \sum_i \frac{|B_i|}{n} \cdot \deg_{B_i} \leq \bar{d}$

  ➢ For large buckets, $\rho_i \leq \frac{|B_i|}{n}(1+\gamma)$, thus $\sum_i \rho_i (1+\beta)^{i-1} \leq \bar{d}(1+\gamma)$

➢ what about underestimating?

Under estimation

- if there were no small bucket:
  - For large buckets, $\rho_i \geq \frac{|B_i|}{n}(1-\gamma)$, thus $\sum_i \rho_i (1+\beta)^{i-1} \geq \sum_i \frac{|B_i|}{n}(1-\gamma)(1+\beta)^{i-1} \geq \frac{1-\gamma}{1+\beta}\sum_i \frac{|B_i|}{n} \cdot (1+\beta)^i \geq (1-\gamma)(1-\beta) \cdot \bar{d}$
- Three types of edges:
  - Large-large (both endpoints in large buckets), counted twice
  - Large-small (one endpoint in a large bucket, one in small), counted once
  - Small-small (both endpoints in small buckets), never counted
- How many small-small edges?
  - By Chernoff, for a small bucket i, $\frac{|B_i|}{n} \approx \frac{|S_i|}{n}$ thus $|B_i| \leq \sqrt{\frac{\epsilon}{n} \cdot \frac{2n}{ct}} = \frac{2\sqrt{\epsilon n}}{ct}$

Under estimation

- if there were no small bucket:
  - For large buckets, $\rho_i \geq \frac{|B_i|}{n}(1-\gamma)$, thus $\sum_i \rho_i(1+\beta)^{i-1} \geq \sum_i \frac{|B_i|}{n}(1-\gamma)(1+\beta)^{i-1} \geq \frac{1-\gamma}{1+\beta}\sum_i \frac{|B_i|}{n} \cdot (1+\beta)^i \geq (1-\gamma)(1-\beta) \cdot \bar{d}$
- Three types of edges:
  - Large-large (both endpoints in large buckets), counted twice
  - Large-small (one endpoint in a large bucket, one in small), counted once
  - Small-small (both endpoints in small buckets), never counted
- How many small-small edges?
  - By Chernoff, for a small bucket i, $|B_i| \leq \sqrt{\frac{\epsilon}{n} \cdot \frac{2n}{ct}} = \frac{2\sqrt{\epsilon n}}{ct}$
  - Thus the total number of such edges is at most $\left(\frac{t \cdot 2\sqrt{\epsilon n}}{ct}\right)^2 \leq O\left(\frac{\epsilon n}{c^2}\right) = O(\epsilon n)$
  - We can ignore them as long as the graph has total degree $\Omega(n)$, i.e., gives $(1+\epsilon)$ multiplicative approximation
- Overall approximation: $2+\epsilon$

Further improvement

- Need to estimate the fraction of large-small edges and account for them
- How? Randomly sample few edges and check if it is large-small
- How to implement such an oracle? Approximate it
    - Pick a random node v in a bucket B_i
    - Pick a random neighbor of v
    - Works if all vertices have the same degree!
    - Approximately works inside the buckets!
    - α_i ≔ fraction of large-small edges
    - Repeat the above to get good probability
- Output of the algorithm $\sum_i \rho_i \boldsymbol{(1+\alpha_i)}(1+\beta)^{i-1}$

Final Algorithm

Algorithm

- Sample a set of vertices S
- Let $S_i = B_i \cap S$ be the samples from the ith bucket
- If $|S_i| \geq \sqrt{\frac{\epsilon}{n}} \cdot \frac{|S|}{c \cdot t}$
    - $\rho_i = \frac{|S_i|}{|S|}$
    - For all $v \in S_i$,
        - Pick a random neighbor u of v
        - $X_v = \begin{cases} 1, & u \text{ is small} \\ 0, & \text{otherwise} \end{cases}$
    - $\alpha_i = \frac{\sum_v X_v}{|S_i|}$
- Output $\sum_i \rho_i (1 + \alpha_i)(1 + \beta)^{i-1}$