Now about the Big Data

- Big data is everywhere
  - Walmart: over 2.5 petabytes of sales transactions
  - Google: an index of over 19 billion web pages
  - Facebook: over 40 billion of pictures
  - …
Now about the Big Data

- Big data is everywhere
  - Walmart*: over 2.5 petabytes of sales transactions
  - Google: an index of over 19 billion web pages
  - Facebook: over 40 billion of pictures
  - ...

- Magazine covers

  Nature ’06  Nature ’08  CACM ’08  Economist ’10
Source and Challenge

Source

- Retailer databases
- Logistics, financial & health data
- Social network
- Pictures by mobile devices
- Internet of Things
- New forms of scientific data
Source and Challenge

- **Source**
  - Retailer databases
  - Logistics, financial & health data
  - Social network
  - Pictures by mobile devices
  - Internet of Things
  - New forms of scientific data

- **Challenge**
  - Volume
  - Velocity
  - Variety (Documents, Stock records, Personal profiles, Photographs, Audio & Video, 3D models, Location data, ...)
Source and Challenge

- **Source**
  - Retailer databases
  - Logistics, financial & health data
  - Social network
  - Pictures by mobile devices
  - Internet of Things
  - New forms of scientific data

- **Challenge**
  - Volume
  - Velocity
  - Variety (Documents, Stock records, Personal profiles, Photographs, Audio & Video, 3D models, Location data, ...)
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, ...
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, …
- The data is stored there, but no time to read them all. What can we do?
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, 

- The data is stored there, but no time to read them all. What can we do?
  - Read some of them. Sublinear in time
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, …
- The data is stored there, but no time to read them all. What can we do?
  - Read some of them. Sublinear in time
- The data is too big to fit in main memory. What can we do?
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, …
- The data is stored there, but no time to read them all. What can we do?
  - Read some of them. Sublinear in time
- The data is too big to fit in main memory. What can we do?
  - Store on the disk (page/block access) Sublinear in I/O
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, 

- The data is stored there, but no time to read them all. What can we do?
  - Read some of them. Sublinear in time

- The data is too big to fit in main memory. What can we do?
  - Store on the disk (page/block access) Sublinear in I/O
  - Throw some of them away. Sublinear in space
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, ...  
- The data is stored there, but no time to read them all. What can we do?
  - Read some of them. Sublinear in time
- The data is too big to fit in main memory. What can we do?
  - Store on the disk (page/block access) Sublinear in I/O
  - Throw some of them away. Sublinear in space
- The data is too big to be stored in a single machine. What can we do if we do not want to throw them away?
What is the meaning of Big Data IN THEORY?

- We don’t define Big Data in terms of TB, PB, EB, . . .

- The data is stored there, but no time to read them all. What can we do?
  - Read some of them. \(\text{Sublinear in time}\)

- The data is too big to fit in main memory. What can we do?
  - Store on the disk (page/block access) \(\text{Sublinear in I/O}\)
  - Throw some of them away. \(\text{Sublinear in space}\)

- The data is too big to be stored in a single machine. What can we do if we do not want to throw them away?
  - Store in multiple machines, which collaborate via communication \(\text{Sublinear in communication}\)
What do we mean by “sublinear”? 

Time/space/communication spent is $o(\text{input size})$
Concretely, theory folks talk about the following ... 

- Sublinear time algorithms 
  - Sublinear time approximation algorithms 
  - Property testing (leave to the future)
Concretely, theory folks talk about the following ...

- **Sublinear time algorithms**
  - Sublinear time approximation algorithms
  - Property testing (leave to the future)

- **Sublinear space algorithms**
  - Data stream algorithms
Concretely, theory folks talk about the following …

- **Sublinear time algorithms**
  - Sublinear time approximation algorithms
  - Property testing (leave to the future)

- **Sublinear space algorithms**
  - Data stream algorithms

- **Sublinear communication algorithms**
  - Multiparty communication protocols/algorithms (special case: MapReduce, BSP, …)
Concretely, theory folks talk about the following …

- Sublinear time algorithms
  - Sublinear time approximation algorithms
  - Property testing (leave to the future)

- Sublinear space algorithms
  - Data stream algorithms

- Sublinear communication algorithms
  - Multiparty communication protocols/algorithms
    (special case: MapReduce, BSP, …)

- Sublinear I/O algorithms (leave to the future)
  - External memory data structures/algorithms
Given a social network graph, we want to compute its average degree. (i.e., the average # of friends people have in the network)

Can we do it without querying the degrees of all nodes? (i.e., asking everyone)
Computing exact average degree is impossible without querying at least \( n - 1 \) nodes (\( n: \# \text{ total nodes} \)).

So our goal is to get a \((1 + \epsilon)\)-approximation w.h.p. (\( \epsilon \) is a very small constant, e.g., 0.01)
Computing exact average degree is impossible without querying at least $n - 1$ nodes ($n$: \# total nodes).

So our goal is to get a $(1 + \epsilon)$-approximation w.h.p. ($\epsilon$ is a very small constant, e.g., 0.01)

Can we simply use sampling?
Why hard? You can’t see everything in sublinear time!

- Computing exact average degree is impossible without querying at least $n - 1$ nodes ($n$: # total nodes).

So our goal is to get a $(1 + \epsilon)$-approximation w.h.p. ($\epsilon$ is a very small constant, e.g., 0.01)

- Can we simply use sampling?

No, it doesn’t work. Consider the star, with degree sequence $(n - 1, 1, \ldots, 1)$. 

![Diagram of a star graph with degree sequence $(n - 1, 1, \ldots, 1)$]
Why hard? You can’t see everything in sublinear time!

- Computing exact average degree is impossible without querying at least $n - 1$ nodes ($n$: \# total nodes).

  So our goal is to get a $(1 + \epsilon)$-approximation w.h.p. ($\epsilon$ is a very small constant, e.g., 0.01)

- Can we simply use sampling?

  No, it doesn’t work. Consider the star, with degree sequence $(n - 1, 1, \ldots, 1)$.

- So can we do anything non-trivial?
  (think about it, and we will discuss later in the course)
Sublinear in space

- **The data stream model** (Alon, Matias and Szegedy 1996)
The data stream model (Alon, Matias and Szegedy 1996)

Applications

- Internet Router.
- Stock data, ad auction, flight logs on tapes, etc.

The router wants to maintain some statistics on data. E.g., want to detect anomalies for security.
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  ![52](image-url)
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  45
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  18
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

![Number 23]
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  ![17]
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

41
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  33
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

29
Why hard? You do see everything but then “forget”!

- **Game 1:** A sequence of numbers

49
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

![12]
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  35
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?
Why hard? You do see everything but then “forget”!

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Bob become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?
  
  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Carol and Eva become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Eva and Bob become friends

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Paul become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Paul become friends

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Eva and Bob unfriends

- **Game 1:** A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2:** Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Alice and Dave become friends

- **Game 1**: A sequence of numbers

  Q: What’s the *median*?

  A: $33$

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Bob and Paul become friends

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Paul unfriends

- **Game 1:** A sequence of numbers
  
  **Q:** What’s the **median**?

  **A:** 33

- **Game 2:** Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Dave and Carol become friends
Game 1: A sequence of numbers

Q: What’s the median?
A: 33

Game 2: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

Q: Are Eva and Bob connected by friends?

- **Game 1**: A sequence of numbers

  Q: What’s the **median**?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Q: Are Eva and Bob connected by friends?

  A: YES. Eva ⇔ Carol ⇔ Dave ⇔ Alice ⇔ Bob

- **Game 1**: A sequence of numbers

  Q: What’s the median?

  A: 33

- **Game 2**: Relationships between Alice, Bob, Carol, Dave, Eva and Paul

  Q: Are Eva and Bob connected by friends?

  A: YES. Eva ⇔ Carol ⇔ Dave ⇔ Alice ⇔ Bob

- Have to allow approx/randomization given a small memory.
Sublinear in communication

- **The model**

  \[ x_1 = 010011 \]
  \[ x_2 = 110111 \]
  \[ x_3 = 111111 \]
  \[ x_k = 100011 \]

  They want to jointly compute \( f(x_1, x_2, \ldots, x_k) \) (e.g., \( f \) is \( \# \) distinct ele)

  Goal: minimize total bits of communication
Sublinear in communication

- **The model**

\[ x_1 = 010011 \quad x_2 = 111011 \quad x_3 = 111111 \]

\[ x_k = 100011 \]

They want to jointly compute \( f(x_1, x_2, \ldots, x_k) \) (e.g., \( f \) is \# distinct ele)

Goal: minimize total bits of communication

- **Applications**

etc.
Let’s think about the **graph connectivity** problem: $k$ machine each holds a set of edges of a graph. 
Goal: compute whether the graph is connected.
Why hard? You do not have a global view of the data.

Let’s think about the **graph connectivity** problem: 
$k$ machine each holds a set of edges of a graph. 
Goal: compute whether the graph is connected.

A trivial solution: each machine sends a **local spanning tree** to the first machine. Cost $O(kn \log n)$ bits.
Why hard? You do not have a global view of the data.

Let's think about the **graph connectivity** problem: $k$ machine each holds a set of edges of a graph.

Goal: compute whether the graph is connected.

A trivial solution: each machine sends a **local spanning tree** to the first machine. Cost $O(kn \log n)$ bits.

Can we do better, e.g., $o(kn)$ bits of communication?
Let’s think about the **graph connectivity** problem: *k* machine each holds a set of edges of a graph.

Goal: compute whether the graph is connected.

A trivial solution: each machine sends a local spanning tree to the first machine. Cost $O(kn \log n)$ bits.

Can we do better, e.g., $o(kn)$ bits of communication?

What if the graph is **node partitioned** among the *k* machines?

That is, each node is stored in 1 machine with all adjacent edges.
Statistical problems

- Frequency moments $F_p$
  - $F_0$: #distinct elements
  - $F_2$: size of self-join
- Heavy hitters
- Quantile
- Entropy
- ...
Problems

Statistical problems

- Frequency moments $F_p$
  - $F_0$: #distinct elements
  - $F_2$: size of self-join
- Heavy hitters
- Quantile
- Entropy
- ...

Graph problems

- Connectivity
- Bipartiteness
- Counting triangles
- Matching
- Minimum spanning tree
- ...

Problems

Statistical problems

• Frequency moments $F_p$
  $F_0$: #distinct elements
  $F_2$: size of self-join
• Heavy hitters
• Quantile
• Entropy
• ...

Numerical linear algebra

Graph problems

• Connectivity
• Bipartiteness
• Counting triangles
• Matching
• Minimum spanning tree
• ...

• $L_p$ regression
• Low-rank approximation
• ...

Problems

Statistical problems

- Frequency moments $F_p$
  - $F_0$: #distinct elements
  - $F_2$: size of self-join
- Heavy hitters
- Quantile
- Entropy
- ...

Graph problems

- Connectivity
- Bipartiteness
- Counting triangles
- Matching
- Minimum spanning tree
- ...

Numerical linear algebra

- $L_p$ regression
- Low-rank approximation
- ...

Geometry problems

- Clustering
- Earth-Mover Distance
- ...

DB queries

- Conjunctive queries

Strings

- Edit distance
- Longest increasing sequence
A few examples using sublinear space
A toy example: Reservoir Sampling

Tasks: Find a **uniform sample** from a stream of unknown length, can we do it in $O(1)$ space?
A toy example: Reservoir Sampling

**Tasks:** Find a **uniform sample** from a stream of unknown length, can we do it in $O(1)$ space?

**Algorithm:** Store 1-st item. When the $i$-th ($i > 1$) item arrives

- With probability $1/i$, replace the current sample;
- With probability $1 - 1/i$, throw it away.
A toy example: Reservoir Sampling

**Tasks:** Find a **uniform sample** from a stream of unknown length, can we do it in $O(1)$ space?

**Algorithm:** Store 1-st item. When the $i$-th ($i > 1$) item arrives

- With probability $1/i$, replace the current sample;
- With probability $1 - 1/i$, throw it away.

**Correctness:** each item is included in the final sample w.p.

$$\frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \ldots \times \left(1 - \frac{1}{n}\right) = \frac{1}{n} \quad (n: \text{total } \# \text{ items})$$

**Space:** $O(1)$
Maintain a sample for Sliding Windows

**Tasks:** Find a uniform sample from the last $w$ items.
Maintain a sample for Sliding Windows

**Tasks:** Find a uniform sample from the last $w$ items.

**Algorithm:**
- For each $x_i$, we pick a random value $v_i \in (0, 1)$.
- In a window $< x_{j-w+1}, \ldots, x_j >$, return value $x_i$ with smallest $v_i$.
- To do this, maintain the set of all $x_i$ in sliding window whose $v_i$ value is minimal among subsequent values.
Maintain a sample for Sliding Windows

**Tasks:** Find a uniform sample from the **last** \( w \) **items**.

**Algorithm:**
- For each \( x_i \), we pick a **random** value \( v_i \in (0, 1) \).
- In a window \(< x_{j-w+1}, \ldots, x_j >\), return value \( x_i \) with **smallest** \( v_i \).
- To do this, **maintain** the set of all \( x_i \) in sliding window whose \( v_i \) value is minimal among subsequent values.

**Correctness:** Obvious.

**Space** (expected): \( \frac{1}{w} + \frac{1}{(w-1)} + \ldots + \frac{1}{1} = \log w \).
Tentative course plan

Part 0: **Introductions (3 lectures)**
- New models for Big Data
- Interesting problems
- Basic probabilistic tools

Part 1: **Sublinear in space (6 lectures)**
- Distinct elements
- Heavy hitters

Part 2: **Sublinear in communication (4 lectures)**
- Sampling ($\ell_0$ sampling)
- Connectivity
- Min-cut and sparsification

Part 3: **Sublinear in time (3 + *2 lectures)**
- Average degree
- *Minimum spanning tree

Part 4: **Student presentations**

Sept. 16 (Mon) and Sept. 18 (Wed), 2 guest lectures by Prof. Funda Ergun
There is no textbook for the class.

Background on Randomized Algorithms:

- **Probability and Computing**  
  by Mitzenmacher and Upfal  
  (Advanced undergraduate textbook)

- **Randomized Algorithms**  
  by Motwani and Raghavan  
  (Graduate textbook)

- **Concentration of Measure for the Analysis of Randomised Algorithms**  
  by Dubhashi, Panconesi  
  (Advanced reference book)
Surveys

- Sketch Techniques for Approximate Query Processing by Cormode
- Data Streams: Algorithms and Applications by Muthukrishnan

Check course website for more resources
Grading

Assignments 42% : There will be two homework assignments, each with about 3-5 questions. Solutions should be typeset in \texttt{LaTeX} and submitted by email.

Project 58% : The project consists of three components:
1. Write a proposal.
2. Make a presentation.
3. Write a report.
(Details will be posted later)
Assignments 42%: There will be two homework assignments, each with about 3-5 questions. Solutions should be typeset in \texttt{LaTeX} and submitted by email.

Project 58%: The project consists of three components:
1. Write a proposal.
2. Make a presentation.
3. Write a report.
(Details will be posted later)

Most important thing:
Learn something about models / algorithmic techniques / theoretical analysis for Big Data.
Prerequisites

A research-oriented course.
Will be quite mathematical.

One is expected to know:
basics on algorithm design and analysis + basic probability.

e.g., have taken B403 “Introduction to Algorithm Design and Analysis” or equivalent courses.

I will NOT start with things like big-O notations, the definitions of expectation and variance, and hashing. But, please always ask at any time if you don’t understand sth.
Is this a course good for my job hunting in industry?

Yes and No.

Yes, if you get to know some advanced (and easily implementable) algorithms for handling big data, that will certainly help. (e.g., Google interview questions)

But, this is a research-oriented course, and is NOT designed for teaching commercially available techniques.
Frequently asked questions

- **Is this a course good for my job hunting in industry?**

  Yes and No.

  Yes, if you get to know some advanced (and easily implementable) algorithms for handling big data, that will certainly help. (e.g., Google interview questions)

  But, this is a research-oriented course, and is NOT designed for teaching commercially available techniques.

- **I haven’t taken B403 “Introduction to Algorithm Design and Analysis” or equivalent courses. Can I take the course? Or, will this course fit me?**

  Generally speaking, this is an advanced algorithm course. It might be difficult if you do not have enough background. But you can always (and welcome to) have a try.
Summary for today

- We have introduced three types of sublinear algorithms: in time, space, and communication.

- We have talked about why algorithmic design in these models/paradigms are difficult.

- We have given a few examples.

- We have talked about the course plan and assessment.
Think about questions (see below) we have talked about today.

If you have novel solutions, please let me know (typeset in latex and send to me by email), and good solutions will receive up to 10 extra points in the final grade.

1. To estimate the average degree of a graph, can we do something non-trivial (that is, without asking the degree of all nodes)?

2. Can $k$ machines test the connectivity of the graph (of $n$ nodes) using $o(nk)$ bits of communication?
Thank you!
Some examples are borrowed from Andrew McGregor’s course
http://people.cs.umass.edu/~mcgregor/courses/CS711S12/index.html