Algorithms at Scale
(Week 13)

Introduction
Today’s Plan

Introduction (Seth)

Streaming Algorithms
  a. Robin Yu, Sidhant Bansal
  b. Eldon Chung, Qiu Siqi

Sampling / Dimensionality Reduction
  a. Xu Kai, Wang Yue
  b. Giovanni Pagliarini, Corentin Dumery

Cache-efficient Algorithms
  a. Foo Guo Wei, Kuan Wei Heng

Wrap-up (Seth)
Streaming Algorithms

HyperLogLog

Algorithm for counting distinct elements in a stream.

Improves on Flajolet-Martin.

State-of-the-art streaming algorithm, in use today.

Windowed Streams

What if only recent data matters?

Techniques for discarding old data in stream.

Example: average temperature over the last 4 hours.
Graph streaming algorithm.

Techniques for estimating the number of triangles in a graph.

Several different sampling approaches.
### Sampling Algorithms

#### Dimensionality Reduction

**2 dim data**

<table>
<thead>
<tr>
<th>(3, 4)</th>
<th>(2, 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(6, 6)</td>
<td>(8, 4)</td>
</tr>
<tr>
<td>(3, 8)</td>
<td>(5, 2)</td>
</tr>
<tr>
<td>(6, 9)</td>
<td>(2, 4)</td>
</tr>
<tr>
<td>(5, 6)</td>
<td>(7, 1)</td>
</tr>
<tr>
<td>(8, 1)</td>
<td>(3, 5)</td>
</tr>
<tr>
<td>(5, 9)</td>
<td>(7, 5)</td>
</tr>
<tr>
<td>(1, 2)</td>
<td>(2, 4)</td>
</tr>
<tr>
<td>(1, 7)</td>
<td>(6, 9)</td>
</tr>
<tr>
<td>(8, 0)</td>
<td>(9, 1)</td>
</tr>
</tbody>
</table>
A small sample is sufficient to get a pretty good estimate of the data.
Sampling Algorithms

Dimensionality Reduction

d dim data

(3, 0, 0, 0, 1, 0, 0, 1)
(2, 1, 1, 0, 5, 1, 0, 0)
(6, 0, 0, 0, 7, 0, 0, 0)
(8, 0, 0, 0, 0, 0, 0, 0)
(3, 0, 0, 4, 1, 0, 4, 0)
(5, 0, 0, 7, 0, 0, 7, 0)
(6, 0, 0, 0, 5, 0, 0, 4)
(2, 4, 4, 3, 0, 4, 3, 7)
(5, 7, 7, 5, 0, 7, 5, 0)
(7, 0, 0, 7, 0, 0, 7, 0)
(8, 3, 7, 0, 0, 7, 0, 0)
(3, 5, 0, 8, 0, 0, 8, 0)
(5, 7, 1, 3, 0, 1, 3, 4)
(7, 0, 0, 5, 0, 0, 5, 7)
(1, 2, 5, 6, 1, 5, 6, 0)
(2, 8, 0, 2, 4, 0, 2, 7)
(1, 1, 0, 5, 6, 0, 5, 7)
(6, 0, 1, 7, 9, 1, 7, 0)
(8, 0, 4, 8, 0, 4, 8, 8)
(9, 0, 5, 5, 0, 5, 5, 3)
### Sampling Algorithms

#### Dimensionality Reduction

<table>
<thead>
<tr>
<th>d dim data</th>
<th>Sample a couple dimension?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 0, 0, 0, 1, 0, 0, 1)</td>
<td>NO. Does not work.</td>
</tr>
<tr>
<td>(2, 1, 1, 0, 5, 1, 0, 0)</td>
<td></td>
</tr>
<tr>
<td>(6, 0, 0, 0, 7, 0, 0, 0)</td>
<td></td>
</tr>
<tr>
<td>(8, 0, 0, 0, 0, 0, 0, 0)</td>
<td></td>
</tr>
<tr>
<td>(3, 0, 0, 4, 1, 0, 4, 0)</td>
<td></td>
</tr>
<tr>
<td>(5, 0, 0, 7, 0, 0, 7, 0)</td>
<td></td>
</tr>
<tr>
<td>(6, 0, 0, 0, 5, 0, 0, 4)</td>
<td></td>
</tr>
<tr>
<td>(2, 4, 4, 3, 0, 4, 3, 7)</td>
<td></td>
</tr>
<tr>
<td>(5, 7, 7, 5, 0, 7, 5, 0)</td>
<td></td>
</tr>
<tr>
<td>(7, 0, 0, 7, 0, 0, 7, 0)</td>
<td></td>
</tr>
<tr>
<td>(8, 3, 7, 0, 0, 7, 0, 0)</td>
<td></td>
</tr>
<tr>
<td>(3, 5, 0, 8, 0, 0, 8, 0)</td>
<td></td>
</tr>
<tr>
<td>(5, 7, 1, 3, 0, 1, 3, 4)</td>
<td></td>
</tr>
<tr>
<td>(7, 0, 0, 5, 0, 0, 5, 7)</td>
<td></td>
</tr>
<tr>
<td>(1, 2, 5, 6, 1, 5, 6, 0)</td>
<td></td>
</tr>
<tr>
<td>(2, 8, 0, 2, 4, 0, 2, 7)</td>
<td></td>
</tr>
<tr>
<td>(1, 1, 0, 5, 6, 0, 5, 7)</td>
<td></td>
</tr>
<tr>
<td>(6, 0, 1, 7, 9, 1, 7, 0)</td>
<td></td>
</tr>
<tr>
<td>(8, 0, 4, 8, 0, 4, 8, 8)</td>
<td></td>
</tr>
<tr>
<td>(9, 0, 5, 5, 0, 5, 5, 3)</td>
<td></td>
</tr>
</tbody>
</table>
Sampling Algorithms

Dimensionality Reduction

d dim
data

Different way of sampling dimension:

Project to a random lower dimensional subspace.
Principal Component Analysis

Project to a lower dimension.

Maximize the variance.

Johnson-Lindenstrauss (JL) Transform

Project to a lower dimension.

Choose the projection at random.
How do you make the JL transform even faster?

Choose a \textit{sparse} projection matrix.

Precondition the vectors before projecting!
Cache-efficient Algorithms

Write-optimized Data Structures

Buffer tree
- search: $O(\log n)$
- insert: $O\left(\frac{1}{B} \log n\right)$

B-tree
- search: $O(\log_B n)$
- insert: $O(\log_B n)$
Cache-efficient Algorithms

Log-Structured Merge Tree

Write-optimized: writes are faster than reads.

Designed to separately optimize memory and disk accesses.

Popular in practice recently.

Cache-Oblivious Lookahead Array

Write-optimized: writes are faster than reads.

No distinction between disk and memory. B, M are unknown.

Primarily of theoretical interest… so far!
Today’s Plan

Introduction (Seth)

Streaming Algorithms
   a. Robin Yu, Sidhant Bansal
   b. Eldon Chung, Qiu Siqi

Sampling / Dimensionality Reduction
   a. Xu Kai, Wang Yue
   b. Giovanni Pagliarini, Corentin Dumery

Cache-efficient Algorithms
   a. Foo Guo Wei, Kuan Wei Heng

Wrap-up (Seth)
Algorithms at Scale
(Week 13)

Semester Recap
CS5234 Goals

- **Intuition:**
  Insights and ideas to help you design your own algorithms for large-scale problems.

- **Arsenal of tools and techniques**
  Tools and techniques to help you analyze and understand algorithms at scale.
Final Exam

Practical Info:

November 27:
• Time: 1pm
• Location: ???
  (NOT HERE)
• Please double-check date and time and location.

To bring:
• One double-sided sheet of paper.
• Otherwise, nothing.

Topics:

Everything from the semester
• Weeks 1-13.
• Problem sets.
• Tutorial problems.
• Bad jokes.
Sampling & Sublinear Time

Streaming & Sketching

Cache Efficient

Parallel
Key techniques

Examples:
1. Polling
   - Reservoir Sampling
   - Chernoff / Hoeffding bounds
   - PAC learning
   - Data analysis (e.g., median)
   - Graph properties (e.g., approximate MST)

Examples:
2. Approximate graph properties (e.g., connected components)
   - Flajolet-Martin (discrete element counting)
   - Clustering algorithms

Examples:
3. Flajolet-Martin
   - Count-Min Sketch
   - Count Sketch
   - Misra-Gries
   - Push-Pull sketch

Examples:
4. B-trees
   - Cache- Oblivious search trees
     / van Emde Boas layout
   - External memory Lubys
   - External memory BFS

Examples:
5. Buffer trees
   - LSM
   - COLA

Examples:
6. Fork-join algorithms
   - Prefix-Sum
   - Map-Reduce algorithms
   - k-machine algorithms
Sublinear Time Algorithms

Simple sampling:

Toy example 1: array all 0’s?
• Gap-style question: All 0’s or far from all 0’s?

Toy example 2: Fraction of 1’s?
• Additive ± ε approximation
• Hoeffding Bound

Is the graph connected?
• Gap-style question.
• O(1) time algorithm.
• Correct with probability 2/3.

Sublinear graph algs:

Number of connected components in a graph.
• Additive approximation algorithm.

Weight of MST
• Multiplicative approximation algorithm.

Key Techniques
• Chernoff and Hoeffding Bounds
<table>
<thead>
<tr>
<th>Sketches</th>
<th>Graph Streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misra-Gries:</td>
<td>Connectivity</td>
</tr>
<tr>
<td>• Item frequency</td>
<td>• Is the graph connected?</td>
</tr>
<tr>
<td>• Heavy Hitters</td>
<td>Bipartite</td>
</tr>
<tr>
<td>Flajolet-Martin:</td>
<td>• Is the graph bipartite?</td>
</tr>
<tr>
<td>• Number of distinct elements</td>
<td>MST</td>
</tr>
<tr>
<td>• Median-of-means technique</td>
<td>• Find a minimum spanning tree</td>
</tr>
<tr>
<td>• Chebychev+Chernoff</td>
<td>Spanners</td>
</tr>
<tr>
<td>Count-Min Sketch:</td>
<td>• Find approximate shortest paths</td>
</tr>
<tr>
<td>• Heavy Hitters</td>
<td>Matching</td>
</tr>
<tr>
<td></td>
<td>• Find an (approximate) maximum</td>
</tr>
<tr>
<td></td>
<td>matching.</td>
</tr>
</tbody>
</table>
Today: Clustering and Streaming

k-median clustering
• Find $k$ centers to minimize the average distance to a center.

LP approximation algorithm
• Find $2k$ centers that give a 4-approximation of the optimal clustering.

Streaming
• Find $k$ centers in a stream of points.
• Use a hierarchical scheme to reduce space.

Other clustering problems
Caching

External Memory

External memory model
• How to predict the performance of algorithms?

B-trees
• Efficient searching

Write-optimized data structures
• Buffer trees

Cache-oblivious algorithms
• van Emde Boas memory layout

Graph Algorithms

Breadth-First-Search
• Sorting your graph

MIS
• Luby’s Algorithm
• Cache-efficient implementation

MST
• Connectivity
• Minimum Spanning Tree
# Parallel Algorithms

## Fork-Join

- **Models of Parallelism**
  - Fork-Join model
  - Work and Span
  - Greedy schedulers

- **Algorithms**
  - Sum
  - MergeSort
  - Parallel Sets
  - BFS
  - Prefix-Sum
  - Luby’s

## Map-Reduce

- **Map-Reduce Model**
  - Cluster computing

- **Algorithms**
  - Word count
  - Join

## k-Machine

- **k-Machine Model**
  - Cluster computing

- **Some simple examples**
  - Luby’s
  - Bellman-Ford

### Minimum Spanning Tree

- Basic algorithm
- Fully distributed
- Lower bound
CS5234: Algorithms at Scale

- Sampling & Sublinear Time
- Streaming & Sketching
- Cache Efficient
- Parallel