The Probabilistic Method

Week 11: Applications, Randomized Algorithms

Joshua Brody

CS49/Math59

Fall 2015

Reading Quiz

Which Frequency Moment can be approximated by a randomized algorithm that uses logarithmic space?

(A) F_0

(B) F_2

(C) F_6

(D) F_{11}

(E) Multiple Answers Correct

Reading Quiz

Which Frequency Moment can be approximated by a randomized algorithm that uses logarithmic space?

(A) **F_0**

(B) **F_2**

(C) **F_6**

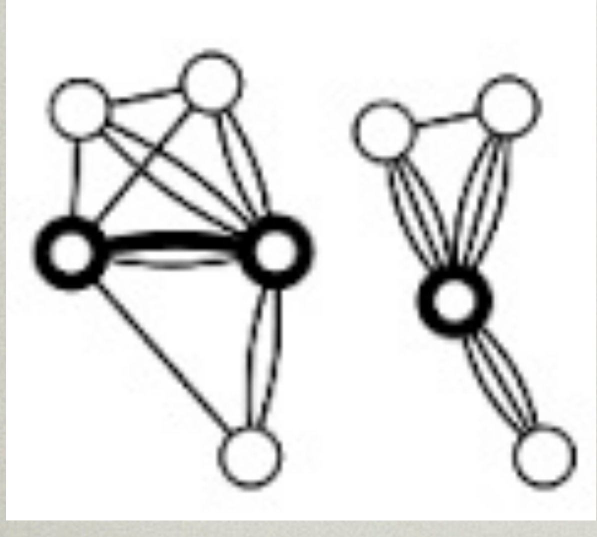
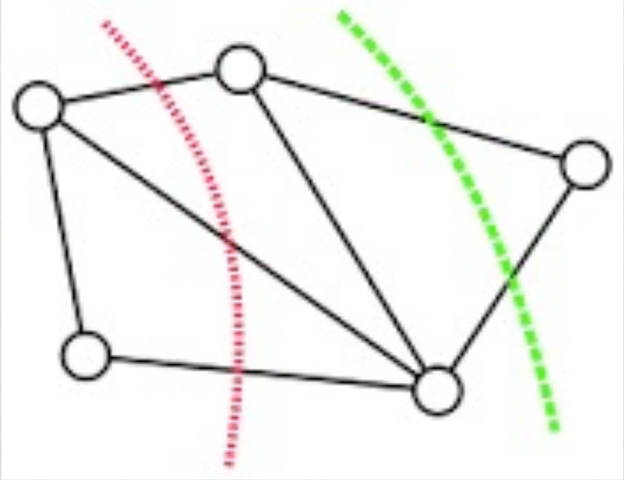
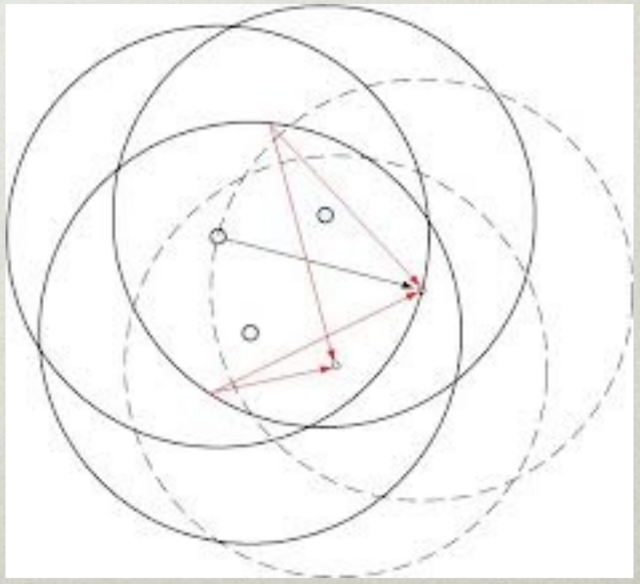
(D) **F_{11}**

(E) **Multiple Answers Correct**

Randomized Algorithms

Types of randomization:

- algorithm uses randomness
  - Las Vegas: *no error, expected runtime*
  - Monte Carlo: *small error, worst-case runtime*
- input is randomized (algorithm often deterministic)

Randomized Algorithms

Why randomized algorithms?

- reduce runtime (or space or ...)
- simpler algorithm
- deterministic not known or impossible

Randomized Algorithms

Why randomized algorithms?

- reduce runtime (or space or ...)
- simpler algorithm
- deterministic not known or impossible

History

- developed late 1970's
- *computational geometry*
- *primality testing* (deterministic algorithm not known until 2003)
- many applications (machine learning, streaming, ...) assume randomness ***by default***

Clicker Question

You see a series of integers $a_1, a_2, \ldots, a_m \in \{1, \ldots, n\}$ that appear one at a time.

Design a randomized algorithm that outputs a_K for a uniform $1 \leq K \leq m$, using as little space as possible.

How much space is used?

(A) $O(m)$

(B) $O(\sqrt{m})$

(C) $O(\log(m))$

(D) $O(\log(\log(m))$

(E) **None of the above**

Clicker Question

You see a series of integers $a_1, a_2, ..., a_m \in \{1, ..., n\}$ that appear one at a time.

Design a randomized algorithm that outputs a_K for a uniform $1 \leq K \leq m$, using as little space as possible.

How much space is used?

(A) $O(m)$

(B) $O(\sqrt{m})$

(C) $O(\log(m))$

(D) $O(\log(\log(m)))$

(E) None of the above

Data Stream Model

Data Stream Model

- *Stream:* m elements from universe of size n
  - *e.g.*, *6,9,4,2,1,3,8,8,6,5,7,2,9,3,3,4,7,12, ...*
- *Goal:* Compute function *F* of stream e.g., # *distinct elements, frequency moments, entropy*, ...
- *Example:* $\mathbf{F_0(S)}$ = number of *distinct elements* in stream *S*

Data Stream Model

- *Stream:* m elements from universe of size n
  - *e.g.*, *6,9,4,2,1,3,8,8,6,5,7,2,9,3,3,4,7,12, ...*
- *Goal:* Compute function *F* of stream e.g., # *distinct elements, frequency moments, entropy*, ...
- *Example:* $\mathbf{F_0(S)}$ = number of *distinct elements* in stream *S*
- Caveats: limited space (e.g., $log(mn)$ or $log^2(mn)$), sequential access to *S*

Data Stream Model

- *Stream:* m elements from universe of size n
  - *e.g.*, *6,9,4,2,1,3,8,8,6,5,7,2,9,3,3,4,7,12, ...*
- *Goal:* Compute function *F* of stream e.g., # *distinct elements, frequency moments, entropy*, ...
- *Example:* **$F_0(S)$** = number of *distinct elements* in stream *S*
- Caveats: limited space (e.g., *$\log(mn)$* or *$\log^2(mn)$*), sequential access to *S*
- *Stream sources:*
  - *Network traffic:* one pass, data not saved is lost forever
  - *Data from hard disk:* multiple passes possible, but costly

Data Stream Model

- Stream: m elements from universe of size n
  - *e.g.*, *6,9,4,2,1,3,8,8,6,5,7,2,9,3,3,4,7,12, ...*
- Goal: Compute function *F* of stream e.g., # *distinct elements, frequency moments, entropy*, ...
- Example: **$F_0(S)$** = number of *distinct elements* in stream *S*
- Caveats: limited space (e.g., *log(mn)* or *$log^2(mn)$*), sequential access to *S*
- Stream sources:
  - Network traffic: one pass, data not saved is lost forever
  - Data from hard disk: multiple passes possible, but costly

Alon, Matias, Szegedy '96 take home messages:
(i) Randomization usually needed
(ii) Approximation usually needed

Data Stream Model

- *Stream:* m elements from universe of size n
  - *e.g.*, *6,9,4,2,1,3,8,8,6,5,7,2,9,3,3,4,7,12, ...*
- *Goal:* Compute function *F* of stream e.g., # *distinct elements, frequency moments, entropy*, ...
- *Example:* $\mathbf{F_0(S)}$ = number of *distinct elements* in stream *S*
- Caveats: limited space (e.g., $log(mn)$ or $log^2(mn)$), sequential access to *S*
- *Stream sources:*
  - *Network traffic:* one pass, data not saved is lost forever
  - *Data from hard disk:* multiple passes possible, but costly

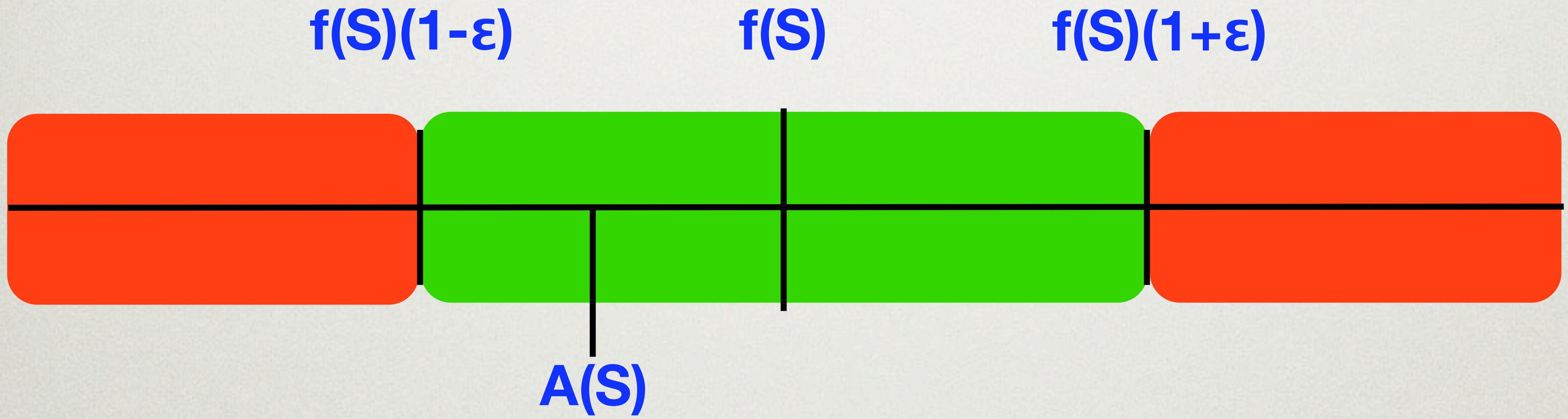
Theorem: [Alon-Matias-Szegedy '96]

Approximating $F_2(s)$ possible in $O(\log(mn)/\varepsilon^2)$ space.

PAC Algorithms

Probably Approximately Correct (PAC) model:

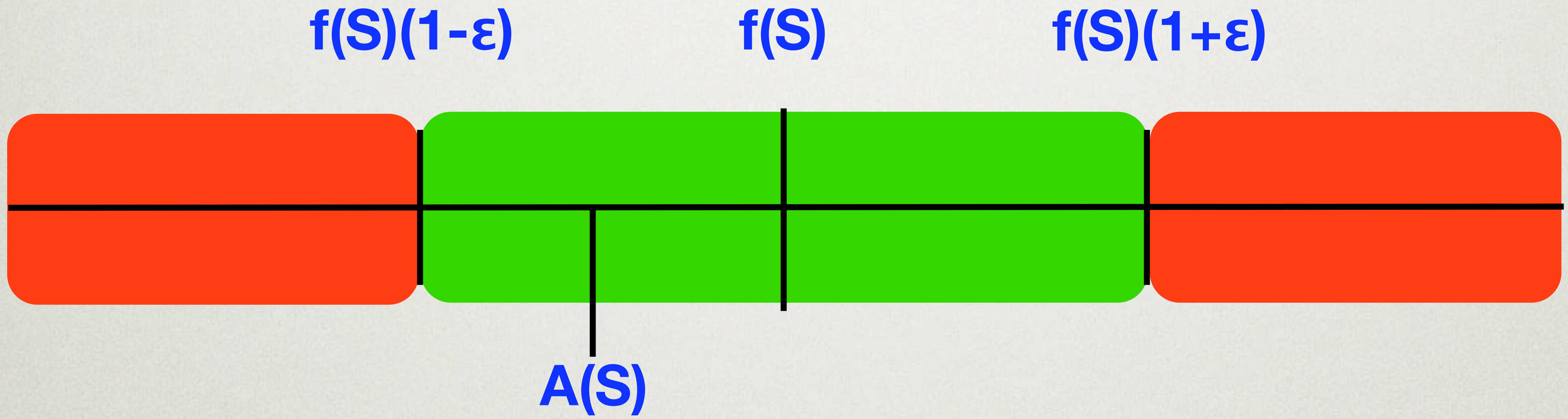
"most of the time we're close enough"



PAC Algorithms

Probably Approximately Correct (PAC) model:

"most of the time we're close enough"



Definition: A streaming algorithm **A** is an (ε,δ)-approximation for **f** if for any stream **S**

$$\Pr[f(s)(1-\varepsilon) \leq A(S) \leq f(s)(1+\varepsilon)] \geq 1-\delta$$

Clicker Question

What is **E[H_j]**?

(A) **E[H_j] = -1**

(B) **E[H_j] = 0**

(C) **E[H_j] = 1/2**

(D) **E[H_j] = f_j**

(E) **None of the above**

Clicker Question

What is $E[H_j]$?

(A) $E[H_j] = -1$

(B) $E[H_j] = 0$

(C) $E[H_j] = 1/2$

(D) $E[H_j] = f_j$

(E) None of the above

Clicker Question

What is **$E[H_i H_j]$**?

(A) $E[H_i H_j] = -1$

(B) $E[H_i H_j] = 0$

(C) $E[H_i H_j] = 1$

(D) It Depends

(E) Multiple Answers Correct

Clicker Question

What is **$E[H_i H_j]$**?

(A) $E[H_i H_j] = -1$

(B) $E[H_i H_j] = 0$

(C) $E[H_i H_j] = 1$

(D) It Depends

(E) Multiple Answers Correct

The Probabilistic Method