

# Advertising on the Web

CS246: Mining Massive Datasets

Jure Leskovec, Stanford University

<http://cs246.stanford.edu>



# Infinite data: Online Algorithms

## High dim. data

Locality  
sensitive  
hashing

Clustering

Dimensional  
ity  
reduction

## Graph data

PageRank,  
SimRank

Community  
Detection

Spam  
Detection

## Infinite data

Filtering  
data  
streams

Web  
advertising

Queries on  
streams

## Machine learning

SVM

Decision  
Trees

Perceptron,  
kNN

## Apps

Recommen  
der systems

Association  
Rules

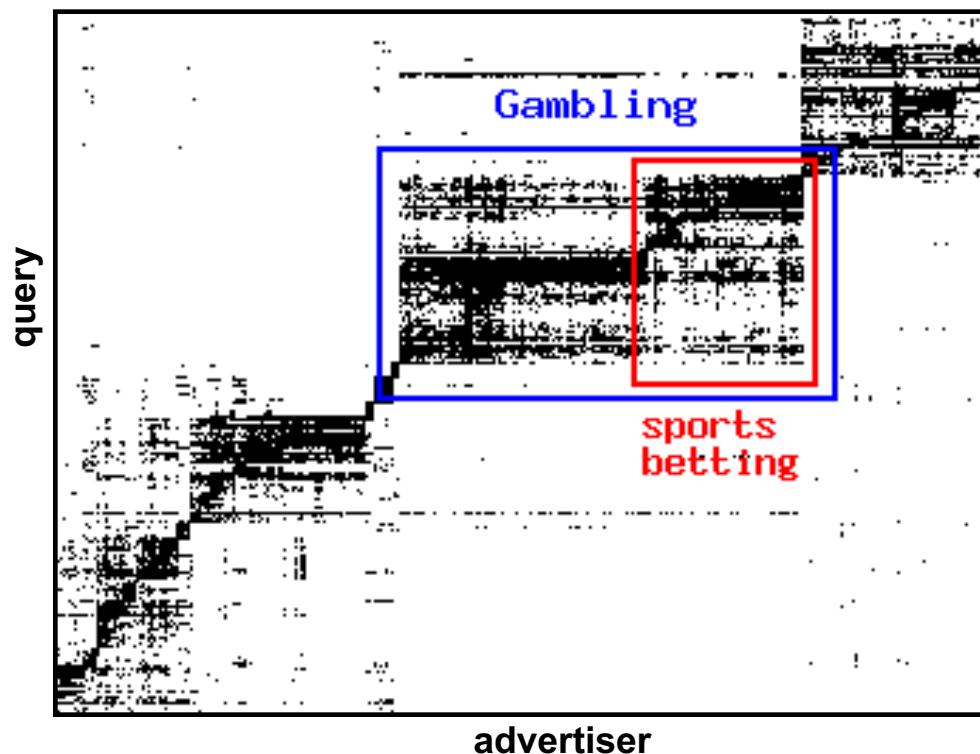
Duplicate  
document  
detection

# Online Algorithms

- **Classic model of algorithms**
  - You get to see the entire input, then compute some function of it
  - In this context, “**offline** algorithm”
- **Online Algorithms**
  - You get to see the input one piece at a time, and need to make irrevocable decisions along the way
  - **Similar to the data stream model**

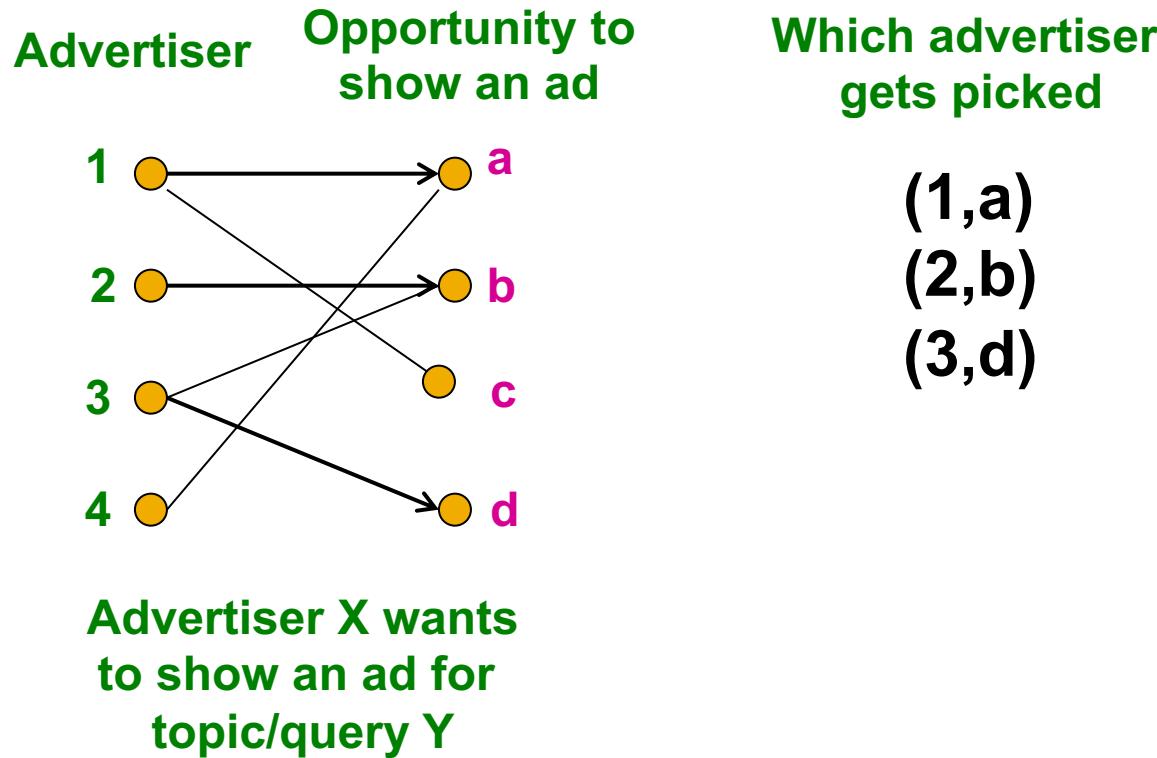
# Sponsored Search: Ads

- Query-to-advertiser graph:



[Andersen, Lang: Communities from seed sets, 2006]

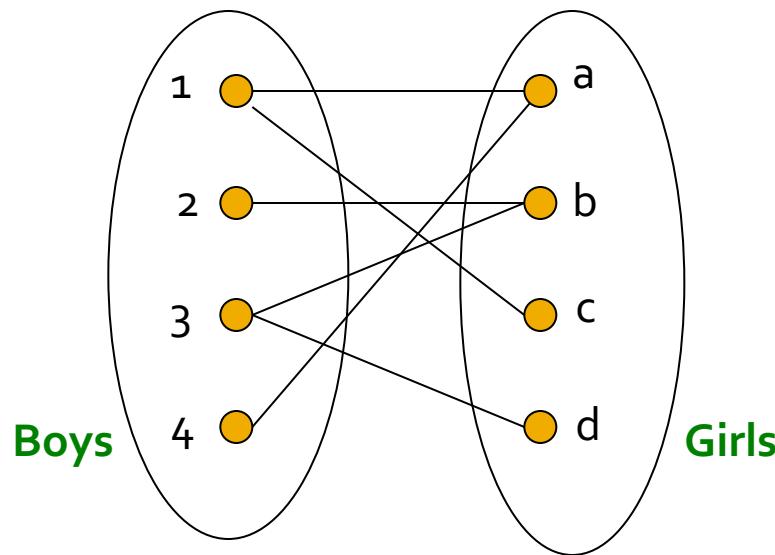
# Graph Matching for Advertising



**This is an online problem:** We have to make decisions as queries/topics show up. We do not know what topics will show up in the future.

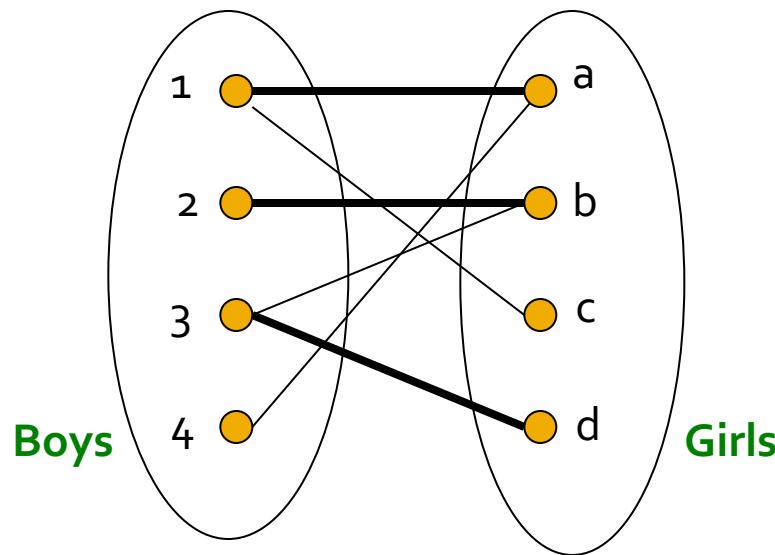
# Online Bipartite Matching

# Example: Bipartite Matching



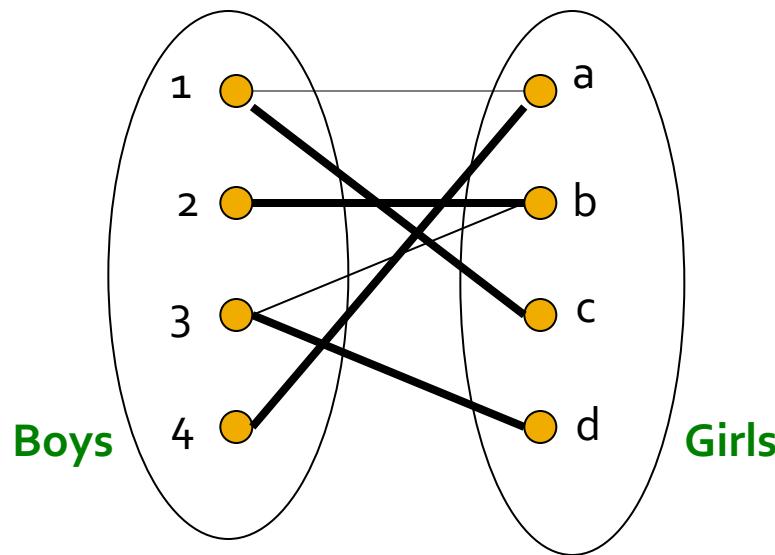
**Nodes: Boys and Girls; Links: Preferences**  
**Goal: Match boys to girls so that the most preferences are satisfied**

# Example: Bipartite Matching



$M = \{(1,a), (2,b), (3,d)\}$  is a **matching**  
**Cardinality of matching =  $|M| = 3$**

# Example: Bipartite Matching



$M = \{(1,c), (2,b), (3,d), (4,a)\}$  is a  
**perfect matching**

**Perfect matching** ... all vertices of the graph are matched

**Maximum matching** ... a matching that contains the largest possible number of matches

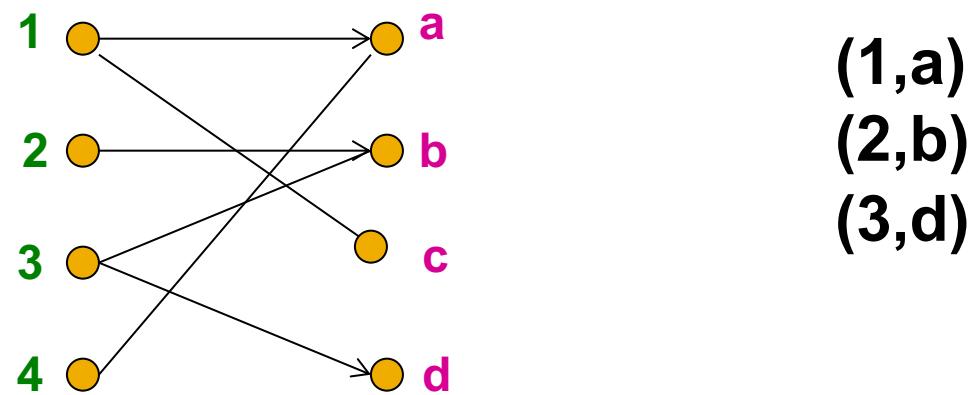
# Matching Algorithm

- **Problem:** Find a maximum matching for a given bipartite graph
  - A perfect one if it exists
- There is a polynomial-time offline algorithm based on augmenting paths (Hopcroft & Karp 1973, see [http://en.wikipedia.org/wiki/Hopcroft-Karp\\_algorithm](http://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm))
- **But what if we do not know the entire graph upfront?**

# Online Graph Matching Problem

- Initially, we are given the set **boys**
- In each **round**, **one girl's choices are revealed**
  - That is, the girl's **edges** are revealed
- **At that time, we have to decide to either:**
  - Pair the **girl** with a **boy**
  - Do not pair the **girl** with any **boy**
- **Example of application:**  
Assigning tasks to servers

# Online Graph Matching: Example



# Greedy Algorithm

- Greedy algorithm for the online graph matching problem:
  - Pair the new girl with **any** eligible boy
    - If there is none, do not pair the girl
- How good is the algorithm?

# Competitive Ratio

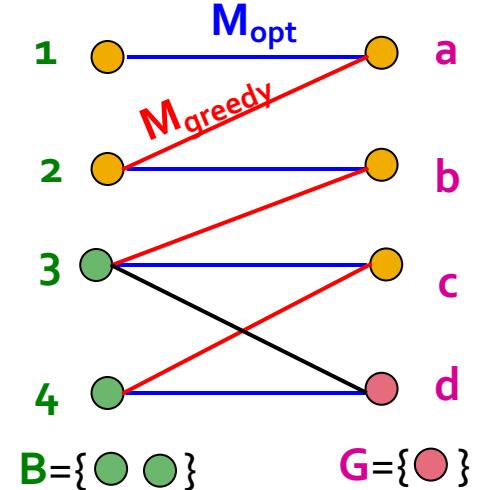
- For input  $I$ , suppose greedy produces matching  $M_{greedy}$  while an optimal matching is  $M_{opt}$

**Competitive ratio =**

$$\min_{\text{all possible inputs } I} (|M_{greedy}| / |M_{opt}|)$$

(what is greedy's worst performance over all possible inputs  $I$ )

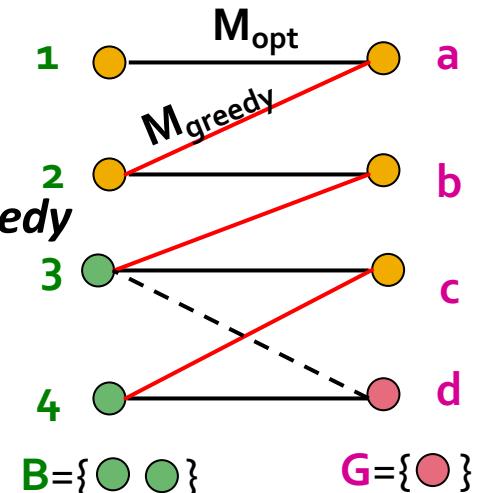
# Analyzing the Greedy Algorithm

- Consider a case:  $M_{greedy} \neq M_{opt}$
  - Consider the set  $G$  of girls matched in  $M_{opt}$  but not in  $M_{greedy}$
  - (1) By definition of  $G$ :  
 $|M_{opt}| \leq |M_{greedy}| + |G|$
  - (2) Define set  $B$  of boys linked to girls in  $G$ 
    - Notice boys in  $B$  are already matched in  $M_{greedy}$ . Why?
      - If there would exist such non-matched (by  $M_{greedy}$ ) boy adjacent to a non-matched girl then greedy would have matched them
- So:  $|M_{greedy}| \geq |B|$  
- 

# Analyzing the Greedy Algorithm

## ■ Summary so far:

- Girls  $\mathbf{G}$  matched in  $M_{opt}$  but not in  $M_{greedy}$
  - Boys  $\mathbf{B}$  adjacent to girls in  $\mathbf{G}$
  - (1)  $|M_{opt}| \leq |M_{greedy}| + |\mathbf{G}|$
  - (2)  $|M_{greedy}| \geq |\mathbf{B}|$
- 
- Optimal matches all girls in  $\mathbf{G}$  to (some) boys in  $\mathbf{B}$
  - (3)  $|\mathbf{G}| \leq |\mathbf{B}|$
- 
- Combining (2) and (3):
  - $|\mathbf{G}| \leq |\mathbf{B}| \leq |M_{greedy}|$



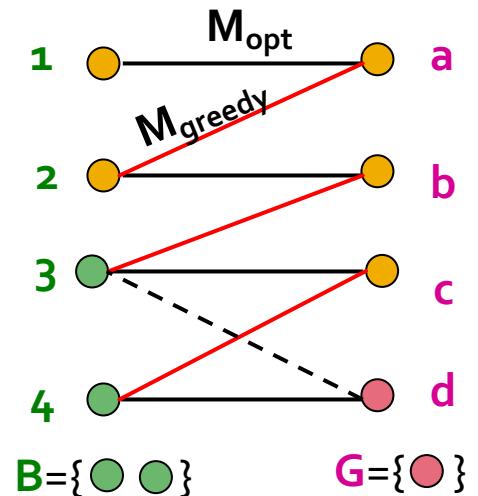
# Analyzing the Greedy Algorithm

- So we have:

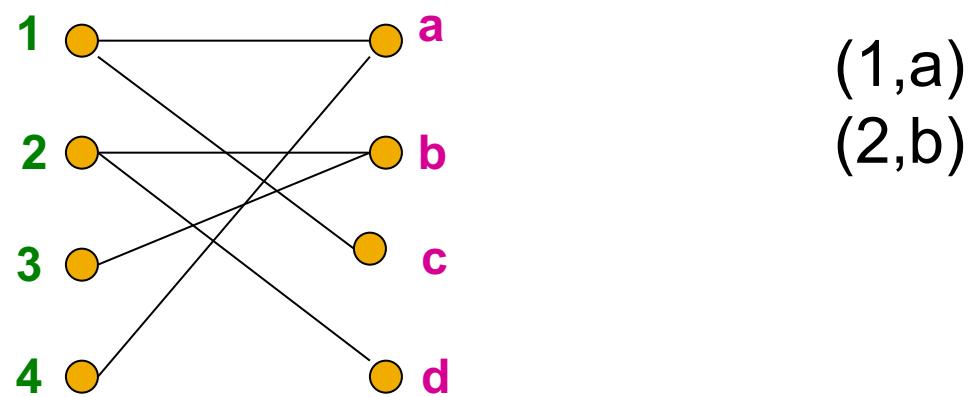
- (1)  $|M_{opt}| \leq |M_{greedy}| + |G|$
- (4)  $|G| \leq |B| \leq |M_{greedy}|$

- Combining (1) and (4):

- Worst case is when  $|G| = |B| = |M_{greedy}|$
- $|M_{opt}| \leq |M_{greedy}| + |M_{greedy}|$
- Then  $|M_{greedy}| / |M_{opt}| \geq 1/2$



# Worst-case Scenario



# Web Advertising

# History of Web Advertising

## ■ Banner ads (1995-2001)

- Initial form of web advertising
- Popular websites charged \$X for every 1,000 “impressions” of the ad
  - Called “**CPM**” rate  
(Cost per thousand impressions)
  - Modeled similar to TV, magazine ads
  - From **untargeted** to **demographically targeted**
  - **Low click-through rates**
    - Low ROI for advertisers



**CPM...cost per mille**  
**Mille...thousand in Latin**

# Performance-based Advertising

- Introduced by Overture around 2000
  - Advertisers **bid on search keywords**
  - When someone searches for that keyword, the **highest bidder's ad is shown**
  - Advertiser is charged only if the ad is clicked on
- Similar model adopted by Google with some changes around 2002
  - Called **Adwords**

# Ads vs. Search Results

## Web

Results 1 - 10 of about 2,230,000 for **geico**. (0.04 sec)

### [GEICO](#) Car Insurance. Get an auto insurance quote and save today ...

GEICO auto insurance, online car insurance quote, motorcycle insurance quote, online insurance sales and service from a leading insurance company.

[www.geico.com/](#) - 21k - Sep 22, 2005 - [Cached](#) - [Similar pages](#)

[Auto Insurance](#) - [Buy Auto Insurance](#)

[Contact Us](#) - [Make a Payment](#)

[More results from www.geico.com >](#)

### [Geico](#), Google Settle Trademark Dispute

The case was resolved out of court, so advertisers are still left without legal guidance on use of trademarks within ads or as keywords.

[www.clickz.com/news/article.php/3547356](#) - 44k - [Cached](#) - [Similar pages](#)

### [Google](#) and [GEICO](#) settle AdWords dispute | The Register

Google and car insurance firm **GEICO** have settled a trade mark dispute over ... Car insurance firm **GEICO** sued both Google and Yahoo! subsidiary Overture in ...

[www.theregister.co.uk/2005/09/09/google\\_geico\\_settlement/](#) - 21k - [Cached](#) - [Similar pages](#)

### [GEICO v. Google](#)

... involving a lawsuit filed by Government Employees Insurance Company (**GEICO**). **GEICO** has filed suit against two major Internet search engine operators, ...

[www.consumeraffairs.com/news04/geico\\_google.html](#) - 19k - [Cached](#) - [Similar pages](#)

## Sponsored Links

### [Great Car Insurance Rates](#)

Simplify Buying Insurance at Safeco  
See Your Rate with an Instant Quote  
[www.Safeco.com](#)

### [Free Insurance Quotes](#)

Fill out one simple form to get  
multiple quotes from local agents.  
[www.HometownQuotes.com](#)

### [5 Free Quotes. 1 Form.](#)

Get 5 Free Quotes In Minutes!  
You Have Nothing To Lose. It's Free  
[sayyessoftware.com/Insurance](#)  
Missouri

# Web 2.0

- **Performance-based advertising works!**
  - Multi-billion-dollar industry
- **Interesting problem:**  
**What ads to show for a given query?**
  - (Today's lecture)
- **If I am an advertiser, which search terms should I bid on and how much should I bid?**
  - (Not focus of today's lecture)

# Adwords Problem

- A stream of queries arrives at the search engine:  $q_1, q_2, \dots$
- Several advertisers bid on each query
- When query  $q_i$  arrives, search engine must pick a subset of advertisers whose ads are shown
- **Goal: Maximize search engine's revenues**
  - **Simple solution:** Instead of raw bids, use the “expected revenue per click” (i.e.,  $\text{Bid} * \text{CTR}$ )
- **Clearly we need an online algorithm!**

# The Adwords Innovation

Advertiser	Bid	CTR	Bid * CTR
A	\$1.00	1%	1 cent
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.25 cents

Click through rate      Expected revenue

# The Adwords Innovation

Advertiser	Bid	CTR	Bid * CTR
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.25 cents
A	\$1.00	1%	1 cent

Instead of sorting advertisers by bid, sort by expected revenue

# Limitations of Simple Algorithm

Instead of sorting advertisers by bid, sort by expected revenue

Advertiser	Bid	CTR	Bid * CTR
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.25 cents
A	\$1.00	1%	1 cent

## Challenges:

- CTR of an ad is unknown
- Advertisers have limited budgets and bid on multiple queries

# Complications: Budget

- Two complications:
  - Budget
  - CTR of an ad is unknown

## 1) Budget: Each advertiser has a limited budget

- Search engine guarantees that the advertiser will not be charged more than their daily budget

# Complications: CTR

- **2) CTR (Click-Through Rate): Each ad-query pair has a different likelihood of being clicked**
  - Advertiser 1 bids \$2, click probability = 0.1
  - Advertiser 2 bids \$1, click probability = 0.5
- **CTR** is predicted or measured historically
  - Averaged over a time period
- **Some complications we will not cover:**
  - **1) CTR is position dependent:**
    - Ad #1 is clicked more than Ad #2

# Complications: CTR

- Some complications we will cover (next lecture):
  - 2) Exploration vs. exploitation
    - Exploit:** Should we keep showing an ad for which we have good estimates of click-through rate?
    - or
    - Explore:** Shall we show a brand new ad to get a better sense of its click-through rate?

# Online Algorithms

## The BALANCE Algorithm

# Adwords Problem

- **Given:**
  - 1. A set of bids by advertisers for search queries
  - 2. A click-through rate for each advertiser-query pair
  - 3. A budget for each advertiser (say for 1 month)
  - 4. A limit on the number of ads to be displayed with each search query
- **Respond to each search query with a set of advertisers such that:**
  - 1. The size of the set is no larger than the limit on the number of ads per query
  - 2. Each advertiser has bid on the search query
  - 3. Each advertiser has enough budget left to pay for the ad if it is clicked upon

# Greedy Algorithm

- Our setting: Simplified environment
  - There is **1** ad shown for each query
  - All advertisers have the same budget  $B$
  - All ads are equally likely to be clicked
  - Bid/value of each ad is the same ( $=1$ )
- Simplest algorithm is greedy:
  - For a query pick any advertiser who has bid **1** for that query
  - Competitive ratio of greedy is  $1/2$

# Bad Scenario for Greedy

# BALANCE Algorithm [MSVV]

- **BALANCE** Algorithm by Mehta, Saberi, Vazirani, and Vazirani
  - For each query, pick the advertiser with the largest unspent budget
    - Break ties arbitrarily (but in a deterministic way)

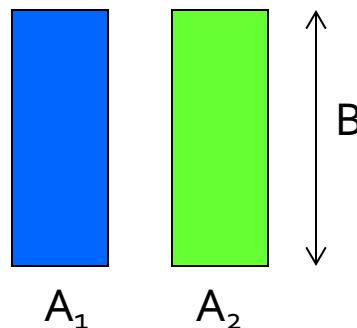
# Example: BALANCE

- Two advertisers A and B
  - A bids on query  $x$ , B bids on  $x$  and  $y$
  - Both have budgets of \$4
- Query stream:  $x \ x \ x \ x \ y \ y \ y \ y$
- BALANCE choice: A B A B B B \_ \_
  - Optimal: A A A A B B B B
- In general: For BALANCE on 2 advertisers  
**Competitive ratio =  $\frac{3}{4}$**

# Analyzing BALANCE

- Consider simple case (w.l.o.g.):
  - 2 advertisers,  $A_1$  and  $A_2$ , each with budget  $B$  ( $\geq 1$ )
  - Optimal solution exhausts both advertisers' budgets
- BALANCE must exhaust at least one budget:
  - If not, we can allocate more queries
    - Whenever BALANCE makes a mistake (both advertisers bid on the query), advertiser's unspent budget only decreases
    - Since optimal exhausts both budgets, one will for sure get exhausted
  - Assume BALANCE exhausts  $A_2$ 's budget, but allocates  $x$  queries fewer than the optimal
    - So revenue of  $BALANCE = 2B - x$  (where OPT is  $2B$ )
  - Let's work out what  $x$  is!

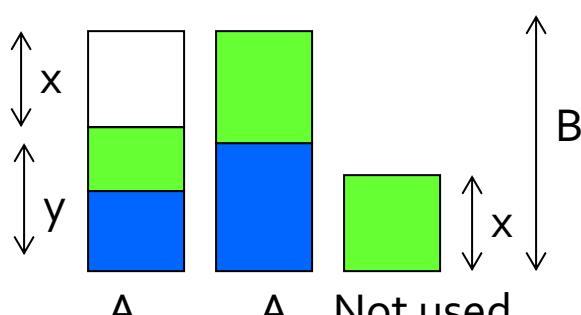
# Analyzing Balance



- Queries allocated to  $A_1$  in optimal solution
- Queries allocated to  $A_2$  in optimal solution

$$\text{Opt revenue} = 2B$$

$$\text{Balance revenue} = 2B - x = B + y$$



**Note:** only green queries can be assigned to neither.  
A blue query could have been assigned to  $A_1$ .

We claim  $y \geq x$  (next slide).

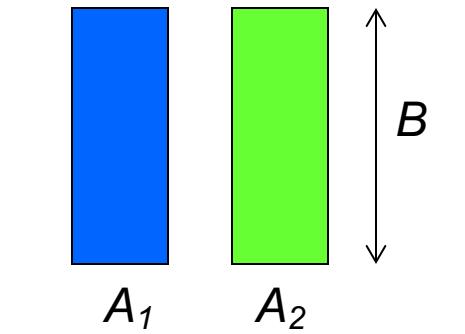
Balance revenue is minimum for  $x=y=B/2$ .

Minimum Balance revenue =  $3B/2$ .

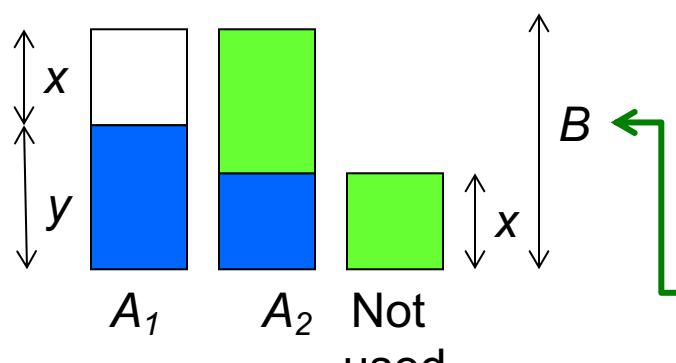
Competitive Ratio =  $3/4$ .

Balance allocation

# Analyzing BALANCE: What's x?



- Queries allocated to  $A_1$  in the optimal solution
- Queries allocated to  $A_2$  in the optimal solution

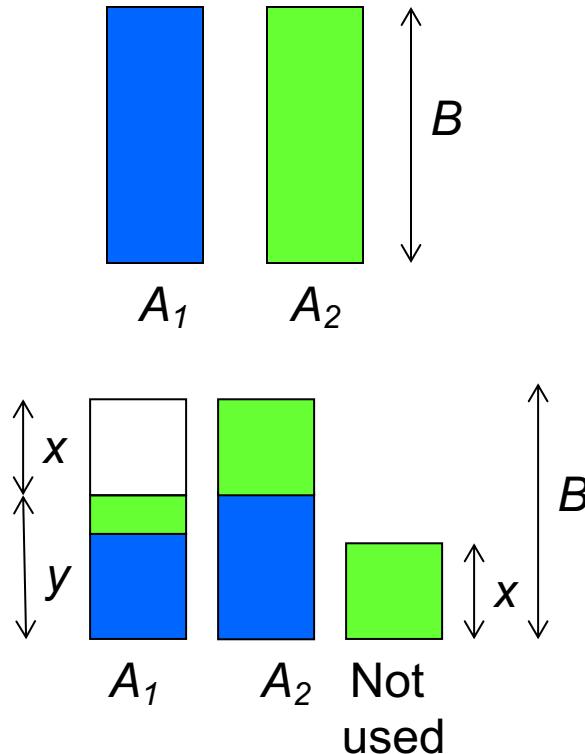


Optimal revenue =  $2B$   
Assume Balance gives revenue =  $2B-x = B+y$   
Assume we exhausted  $A_2$ 's budget

**Notice:** Unassigned queries should be assigned to  $A_2$  (since if we could assign to  $A_1$  we would since we still have the budget)  
**Goal:** Show we have  $y \geq B/2$

**Case 1)** BALANCE assigns at  $\geq B/2$  blue queries to  $A_1$ .  
Then trivially,  $y \geq B/2$

# Analyzing BALANCE: What's x?



- Queries allocated to  $A_1$  in the optimal solution
- Queries allocated to  $A_2$  in the optimal solution

Optimal revenue =  $2B$

Assume Balance gives revenue =  $2B-x = B+y$

Assume we exhausted  $A_2$ 's budget

**Unassigned queries should be assigned to  $A_2$**   
(if we could assign to  $A_1$  we would since we still have the budget)

**Goal: Show we have  $y \geq B/2$**

**Balance revenue is minimum for  $x = y = B/2$**   
Minimum Balance revenue =  $3B/2$   
**Competitive Ratio:  $BAL/OPT = 3/4$**

**Case 2) BALANCE assigns  $\geq B/2$  blue queries to  $A_2$ .**

Consider the last blue query assigned to  $A_2$ .

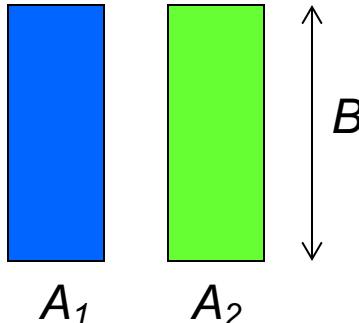
At that time,  $A_2$ 's unspent budget must have been at least as big as  $A_1$ 's.

That means at least as many queries have been assigned to  $A_1$  as to  $A_2$ .

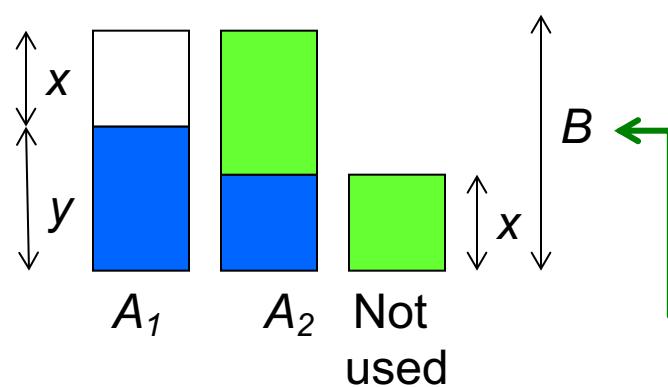
At this point, we have already assigned at least  $B/2$  queries to  $A_2$ .

So,  $x \leq B/2$  and  $x + y = B$  then  $y > B/2$

# Analyzing BALANCE: What's x?



- █ Queries allocated to  $A_1$  in the optimal solution
- █ Queries allocated to  $A_2$  in the optimal solution



Optimal revenue =  $2B$   
 Assume Balance gives revenue =  $2B-x = B+y$   
 Assume we exhausted  $A_2$ 's budget

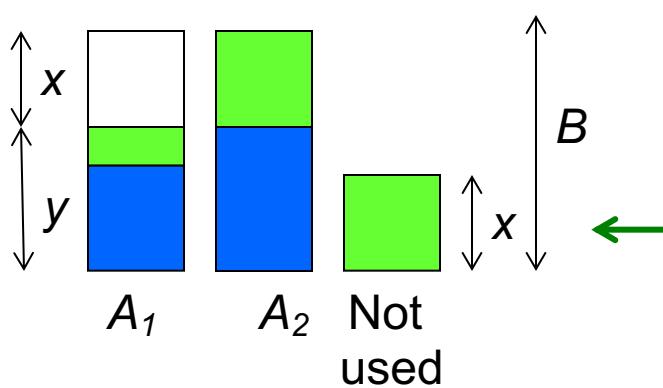
**Unassigned queries should be assigned to  $A_2$**   
 (if we could assign to  $A_1$  we would since we still have the budget)

**Goal:** Show we have  $y \geq x$

**Case 1)**  $\leq \frac{1}{2}$  of  $A_1$ 's queries got assigned to  $A_2$   
 then  $y \geq B/2$

**Case 2)**  $> \frac{1}{2}$  of  $A_1$ 's queries got assigned to  $A_2$   
 then  $x \leq B/2$  and  $x + y = B$

**Balance revenue is minimum for  $x = y = B/2$**   
 Minimum Balance revenue =  $3B/2$   
**Competitive Ratio = 3/4**



# BALANCE: General Result

- In the general case, worst competitive ratio of BALANCE is  $1 - 1/e = \text{approx. } 0.63$ 
  - $e = 2.7182$
  - Interestingly, no online algorithm has a better competitive ratio!
- Let's see the worst case example that gives this ratio

# Worst case for BALANCE

- **$N$  advertisers:**  $A_1, A_2, \dots, A_N$ 
  - Each with budget  $B > N$
- **Queries:**
  - $N \cdot B$  queries appear in  $N$  rounds of  $B$  queries each
- **Bidding:**
  - Round 1 queries: bidders  $A_1, A_2, \dots, A_N$
  - Round 2 queries: bidders  $A_2, A_3, \dots, A_N$
  - Round  $i$  queries: bidders  $A_i, \dots, A_N$
- **Optimum allocation:**

Allocate all round  $i$  queries to  $A_i$ ;

  - Optimum revenue  $N \cdot B$

# BALANCE Allocation



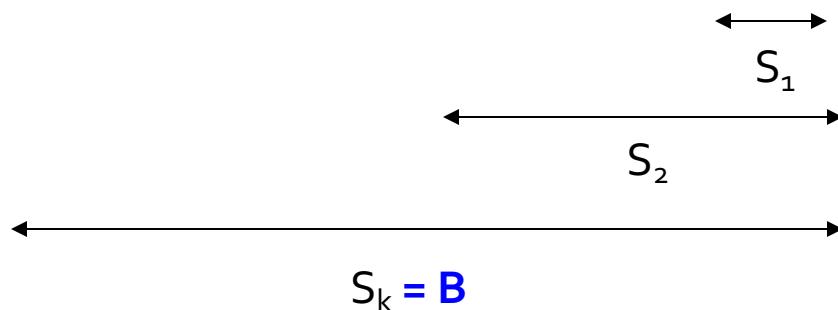
BALANCE assigns each of the queries in round 1 to **N** advertisers. After  $k$  rounds, sum of allocations to each of advertisers  $A_k, \dots, A_N$  is

$$S_k = S_{k+1} = \dots = S_N = \sum_{i=1}^k \frac{B}{N-(i-1)}$$

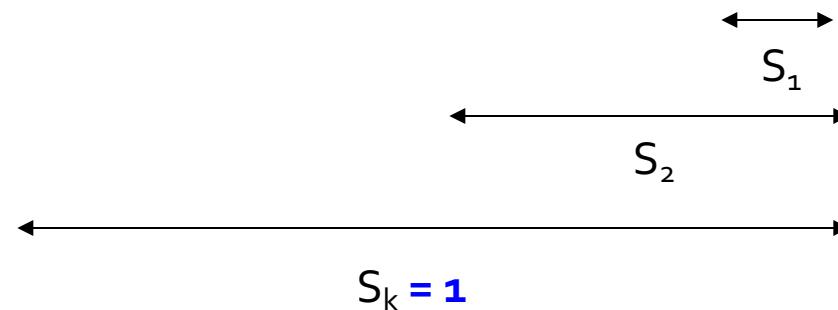
If we find the smallest  $k$  such that  $S_k \geq B$ , then after  $k$  rounds we cannot allocate any queries to any advertiser

# BALANCE: Analysis

$B/1 \quad B/2 \quad B/3 \quad \dots \quad B/(N-(k-1)) \quad \dots \quad B/(N-1) \quad B/N$

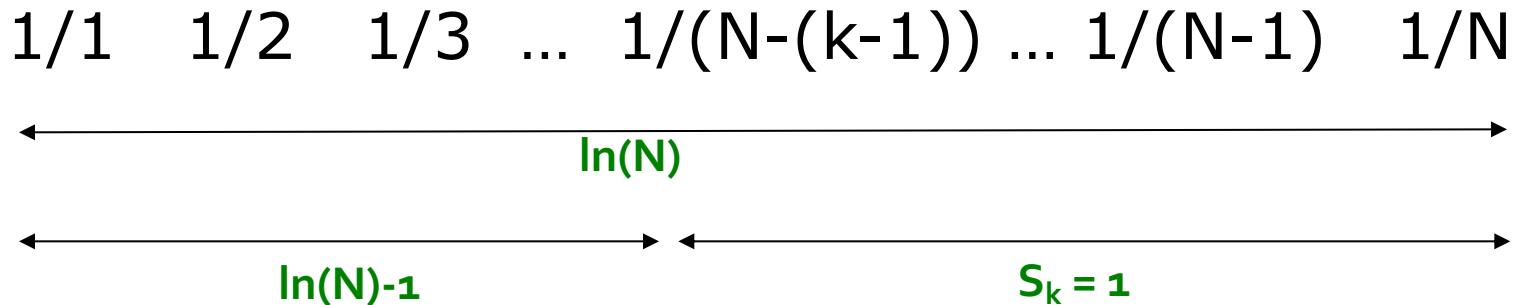


$1/1 \quad 1/2 \quad 1/3 \quad \dots \quad 1/(N-(k-1)) \quad \dots \quad 1/(N-1) \quad 1/N$



# BALANCE: Analysis

- Fact:  $H_n = \sum_{i=1}^n 1/i \approx \ln(n)$  for large  $n$ 
  - Result due to Euler



- $S_k = 1$  implies:  $H_{N-k} = \ln(N) - 1 = \ln\left(\frac{N}{e}\right)$
- We also know:  $H_{N-k} = \ln(N - k)$
- So:  $N - k = \frac{N}{e}$ 
  - $N$  terms sum to  $\ln(N)$ .
  - Last  $k$  terms sum to 1.
  - First  $N-k$  terms sum to  $\ln(N-k)$  but also to  $\ln(N)-1$
- Then:  $k = N\left(1 - \frac{1}{e}\right)$

# BALANCE: Analysis

- So after the first  $k=N(1-1/e)$  rounds, we cannot allocate a query to any advertiser
- **Revenue =  $B \cdot N (1-1/e)$**
- **Competitive ratio =  $1-1/e$**

# General Version of the Problem

- **Arbitrary bids and arbitrary budgets!**
- Consider we have 1 query  $q$ , advertiser  $i$ 
  - Bid =  $x_i$
  - Budget =  $b_i$
- **In a general setting BALANCE can be terrible**
  - Consider two advertisers  $A_1$  and  $A_2$
  - $A_1: x_1 = 1, b_1 = 110$
  - $A_2: x_2 = 10, b_2 = 100$
  - Consider we see 10 instances of  $q$
  - BALANCE always selects  $A_1$  and earns 10
  - Optimal earns 100

# Generalized BALANCE

- **Arbitrary bids:** consider query  $q$ , bidder  $i$ 
  - Bid =  $x_i$
  - Budget =  $b_i$
  - Amount spent so far =  $m_i$
  - Fraction of budget left over  $f_i = 1-m_i/b_i$
  - Define  $\psi_i(q) = x_i(1-e^{-f_i})$
- Allocate query  $q$  to bidder  $i$  with largest value of  $\psi_i(q)$
- **Same competitive ratio ( $1-1/e$ ) = 0.63**