Advertising on the Web

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
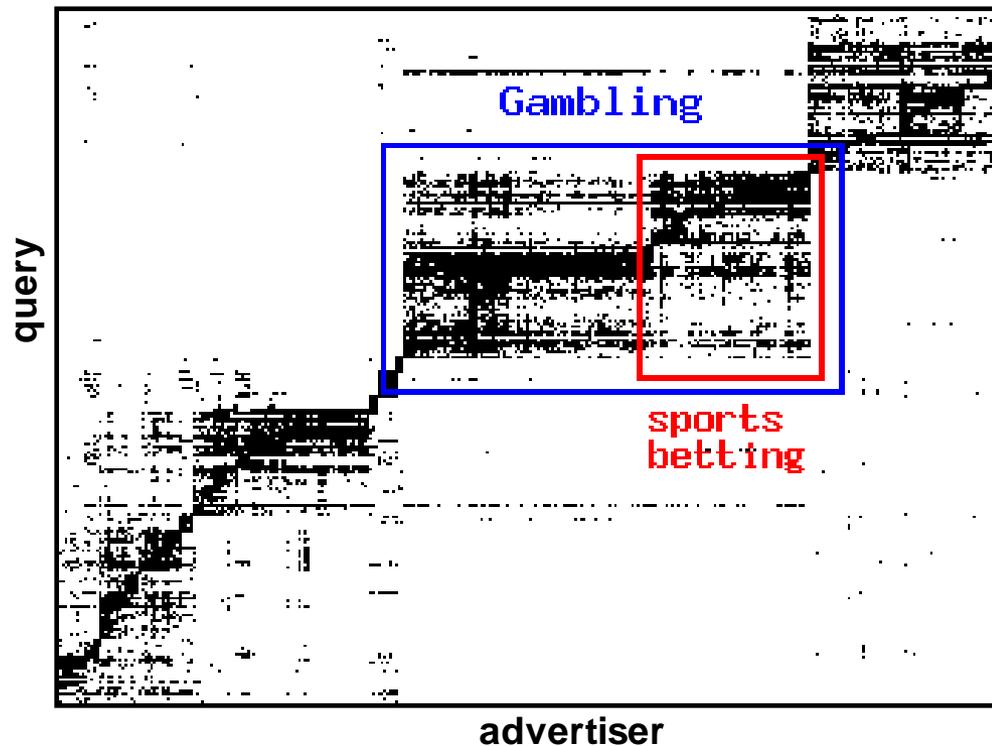Mina Ghashami, Amazon
http://cs246.stanford.edu

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
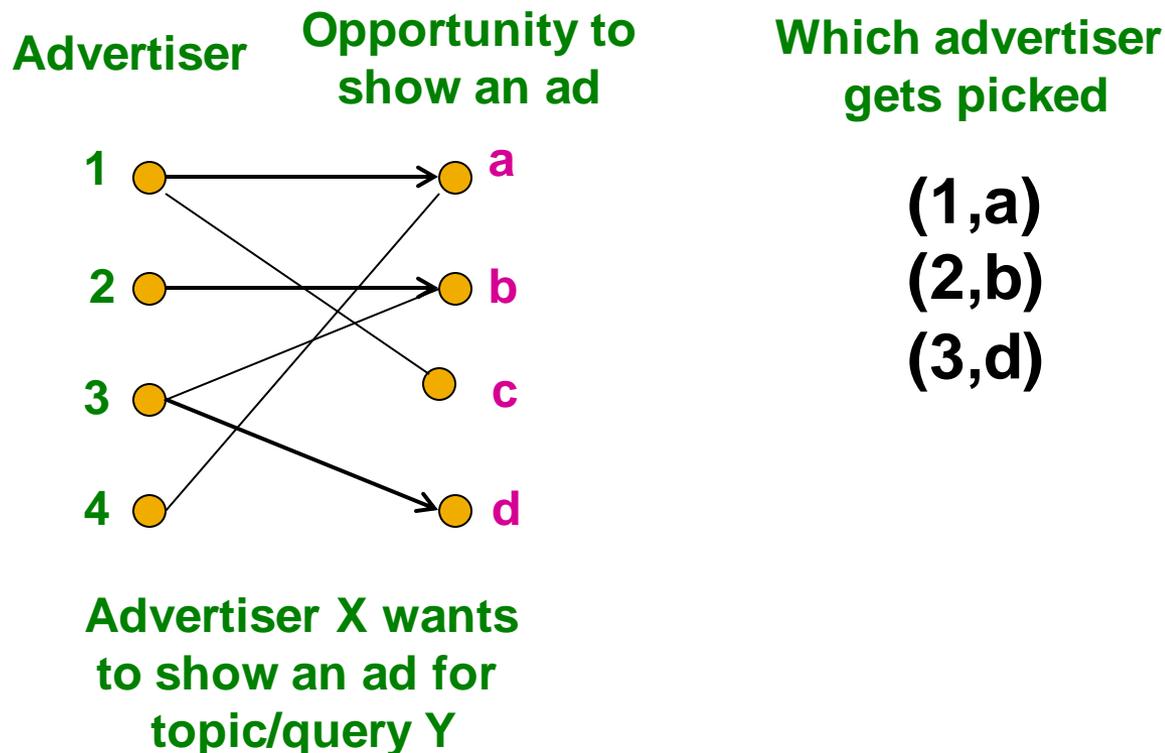
  - **Similar to the data stream model**

Sponsored Search: Ads

- **Query-to-advertiser graph:**

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

Graph Matching for Advertising

Advertiser   **Opportunity to show an ad**   **Which advertiser gets picked**

1 a

2 b

3 c

4 d

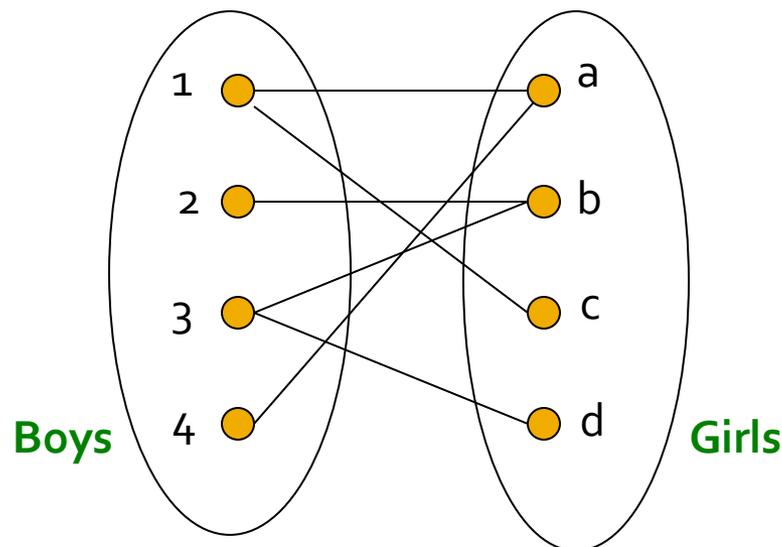**(1,a)**
(2,b)
(3,d)

Advertiser X wants to show an ad for topic/query Y

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.
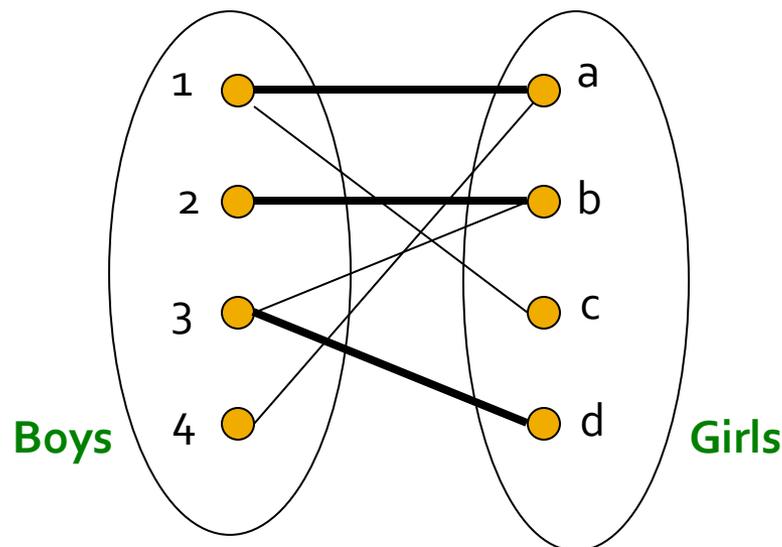
Online Bipartite Matching

Example: Bipartite Matching

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

Note: edges are only preferences with no weight or order.

M = {(1,a),(2,b),(3,d)} is a **matching**

Cardinality of matching = |M| = 3

Matching means that we are not using any vertex twice
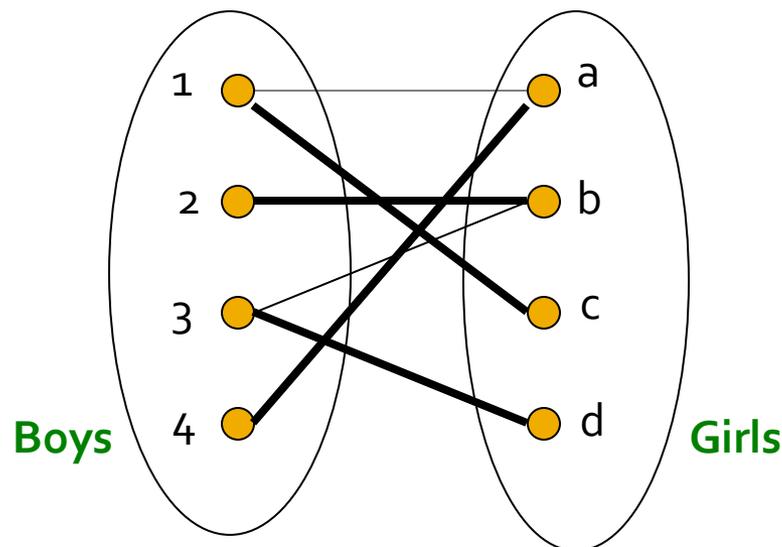
M = {(1,c),(2,b),(3,d),(4,a)} is a
perfect matching

Perfect matching … all vertices of the graph are matched
Maximum matching … 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)

- **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

Note: Matching means that we are not using any girl or boy twice
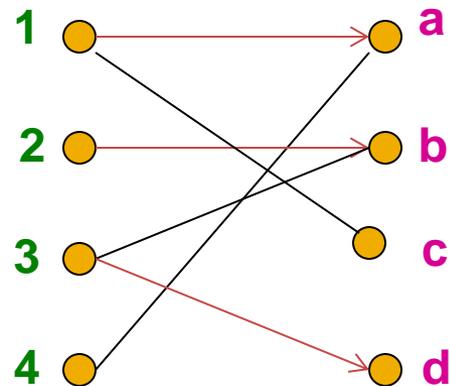
(1,a)
(2,b)
(3,d)

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 =

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

(what is greedy's <u>worst</u> performance <u>over all possible</u> 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**:
$$|M_{opt}| \leq |M_{greedy}| + |G|$$

Edge color indicates matching in M_{opt} (blue) vs. M_{greedy} (red).

- **(2)** Define set **B of boys** <u>linked</u> 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
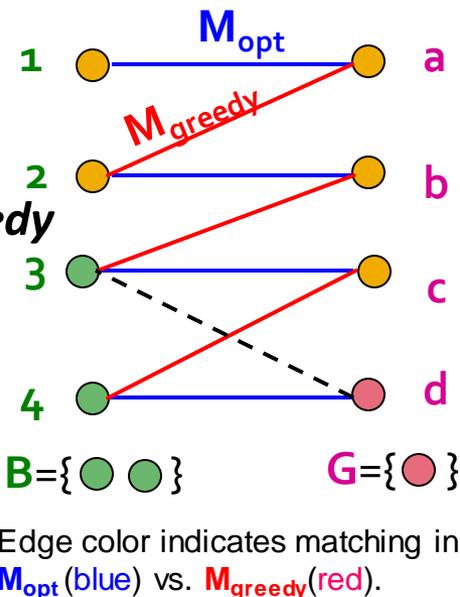
So: $|M_{greedy}| \geq |B|$
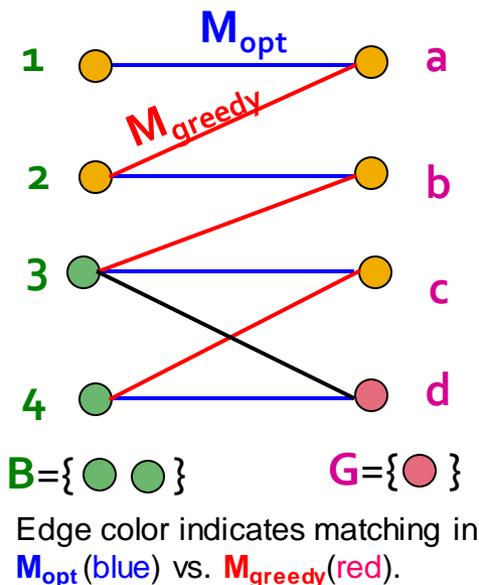
Analyzing the Greedy Algorithm

- **Summary so far:**
  - Girls G matched in M_{opt} but not in M_{greedy}
  - Boys B adjacent to girls in G
  - **(1)** $|M_{opt}| \le |M_{greedy}| + |G|$
  - **(2)** $|M_{greedy}| \ge |B|$

- Optimal matches all girls in G to (some) boys in B
  - **(3)** $|G| \le |B|$

- Combining **(2)** and **(3)**:
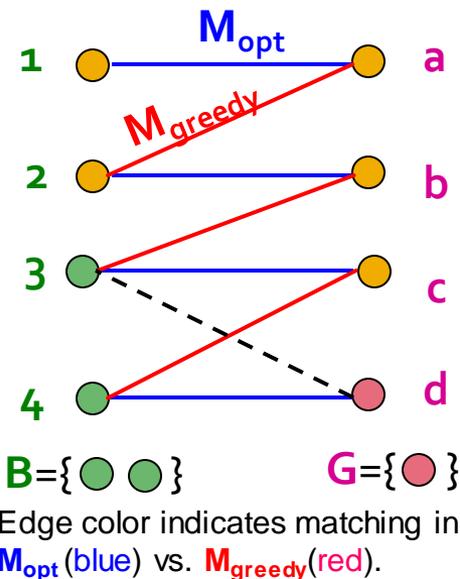 - **(4)** $|G| \le |B| \le |M_{greedy}|$

Edge color indicates matching in M_{opt} (blue) vs. M_{greedy} (red).
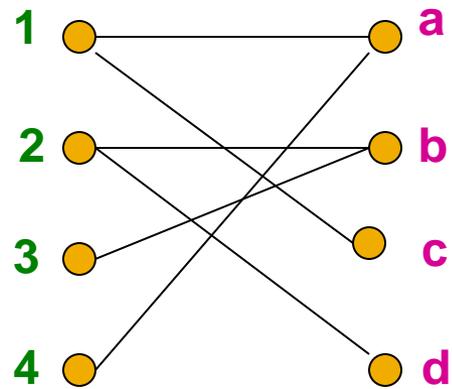
- **So we have:**
  - **(1)** $|M_{opt}| \leq |M_{greedy}| + |G|$
  - **(4)** $|G| \leq |B| \leq |M_{greedy}|$

- **Combining (1) and (4):**
  - Worst case is when $|G| = |B| = |M_{greedy}|$
  - $|M_{opt}| \leq |M_{greedy}| + |M_{greedy}|$
  - **Then** $|M_{greedy}|/|M_{opt}| \geq 1/2$

$B = \{ \bullet \ \bullet \}$ $G = \{ \bullet \}$

Edge color indicates matching in M_{opt} (blue) vs. M_{greedy} (red).

Worst-case Scenario

(1,a)
(2,b)

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 2.0

- **Performance-based advertising works!**
  - Multi-billion-dollar industry

- **Interesting problem:**
  Which 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: $q_1, q_2, \ldots$
- Several advertisers bid on each query
- When query q_i arrives, search engine must pick a subset of advertisers to show their ads

- **Goal: Maximize search engine's revenues**

  - **Simple solution:** Instead of raw bids, use the "**expected revenue per click**" (i.e., **Bid*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 on query A, click probability = 0.1
  - **Advertiser 2** bids $1 on query B, click probability = 0.5
- **CTR** is predicted or measured historically
  - Averaged over a time period

- **Some complications we will <u>not</u> 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

- **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***

  - Worst case greedy choice: *B B B B* _ _ _ _

  - Optimal: **A A A A B B B B**

  - **Competitive ratio = ½**

- **This is the worst case!**

  - **Note:** Greedy algorithm is deterministic – it always resolves draws in the same way

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 = ¾**

Analyzing BALANCE

- ## Consider simple case (w.l.o.g.):
  - **2** advertisers, A_1 and A_2, each with budget **B** (≥ 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)**
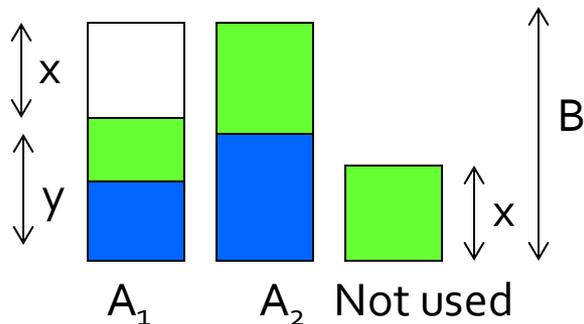 - **Let's work out what x is!**

Analyzing Balance
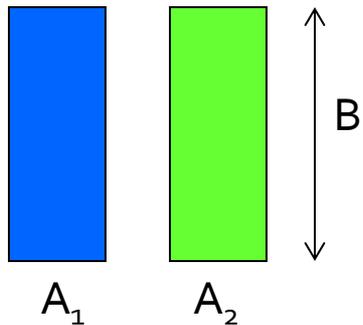
■ Queries allocated to A_1 in optimal solution

■ Queries allocated to A_2 in optimal solution
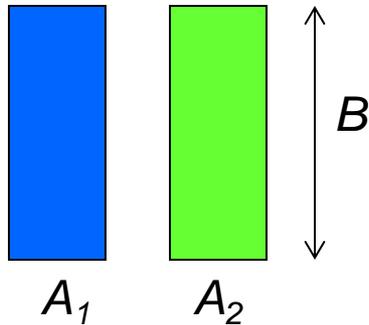
Opt revenue = 2B

Balance revenue = 2B-x = B+y

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 $\geq B/2$ blue queries to A$_1$.
Then trivially, $y \geq B/2$
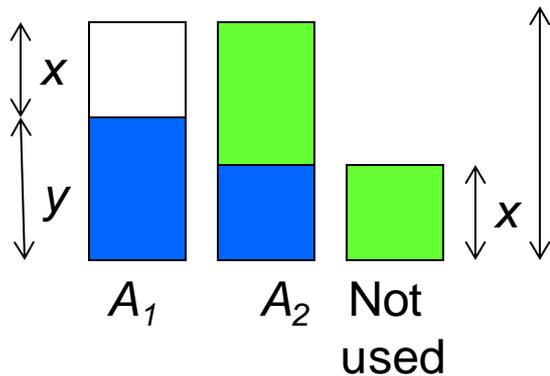
Analyzing BALANCE: What's x?

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**
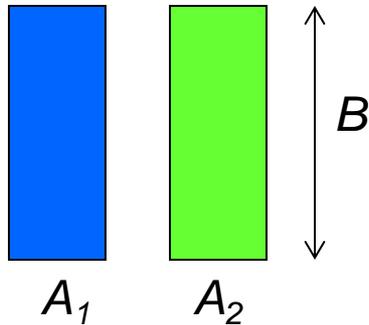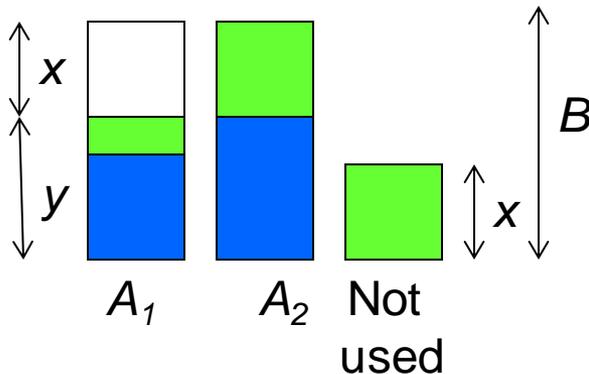
Assume we exhausted A_2's budget

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 ≥ 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 ≥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** $y > B/2$

BALANCE: General Result

- **In the general case, worst competitive ratio of BALANCE is 1–1/e = 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: _A₁, A₂, … A_N_**
  - Each with budget **_B_ > _N_**
- **Queries:**
  - **_N·B_** queries appear in **_N_** rounds of **_B_** queries each
- **Bidding:**
  - Round **1** queries: bidders $A_1, A_2,$ …, A_N
  - Round **2** queries: bidders $A_2, A_3,$ …, A_N
  - Round **_i_** queries: bidders $A_i,$ …, A_N
- **Optimum allocation:**
  Allocate all round **_i_** queries to **_A_i_**
  - Optimum revenue **_N·B_**

BALANCE Allocation
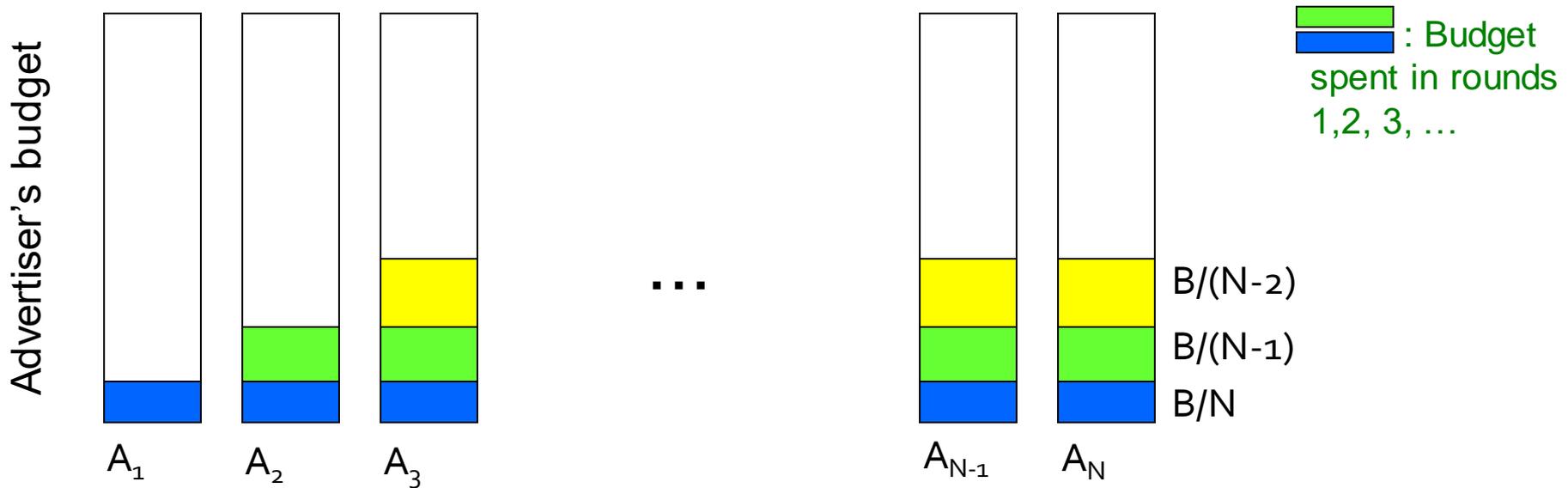
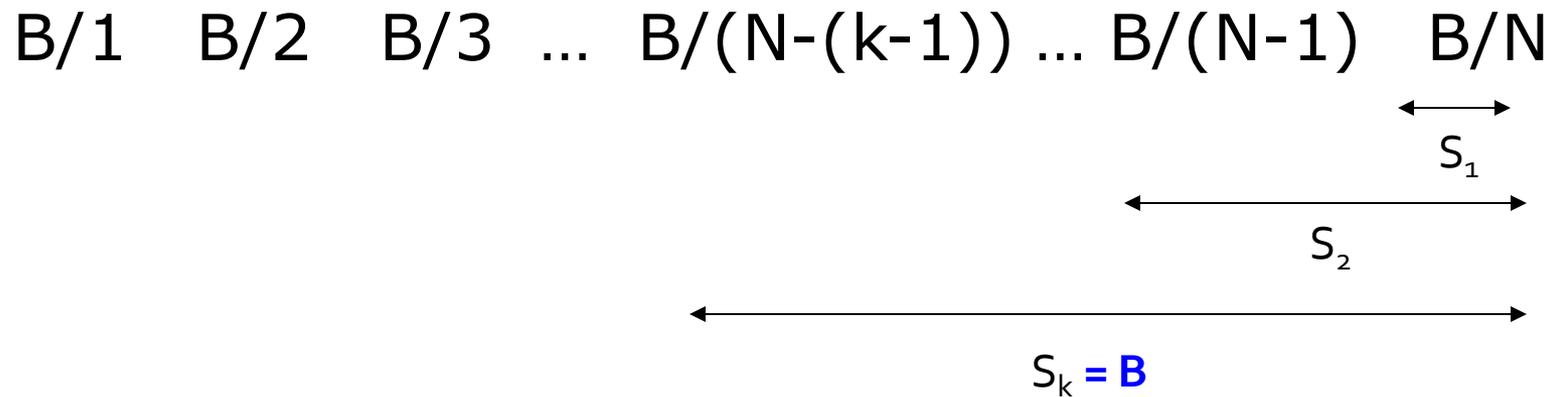BALANCE assigns each of the queries in round 1 to **N** advertisers. After **k** rounds, sum of allocations S_k to each of advertisers

$$A_k,...,A_N \text{ is } S_k = S_{k+1} = \cdots = S_N = \sum_{i=1}^{k} \frac{B}{N-(i-1)}$$

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

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

S_1

S_2

$S_k = B$

Can divide everything by B:

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

S_1

S_2

$S_k = 1$
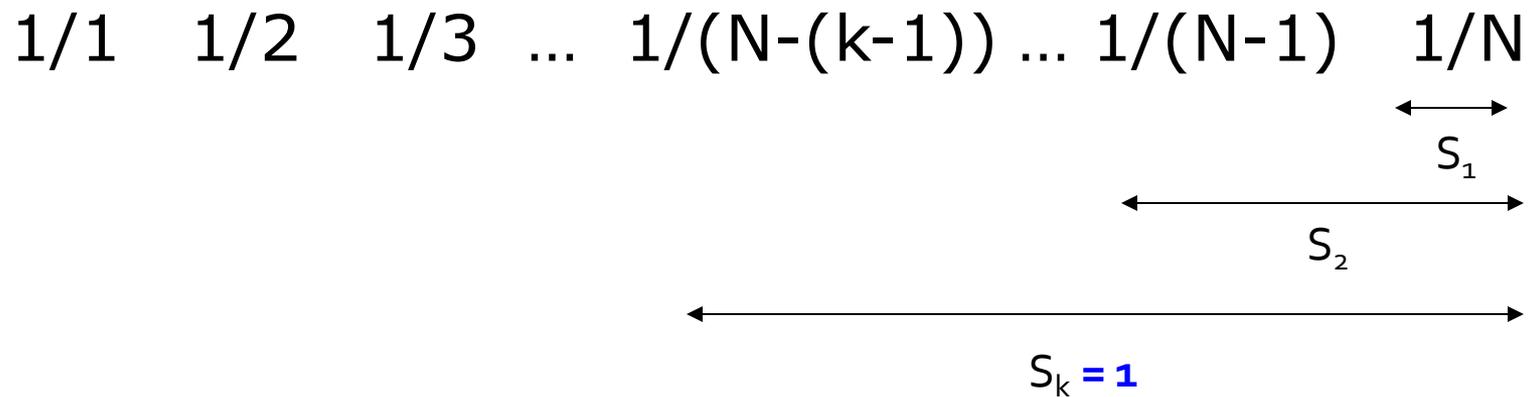
BALANCE: Analysis

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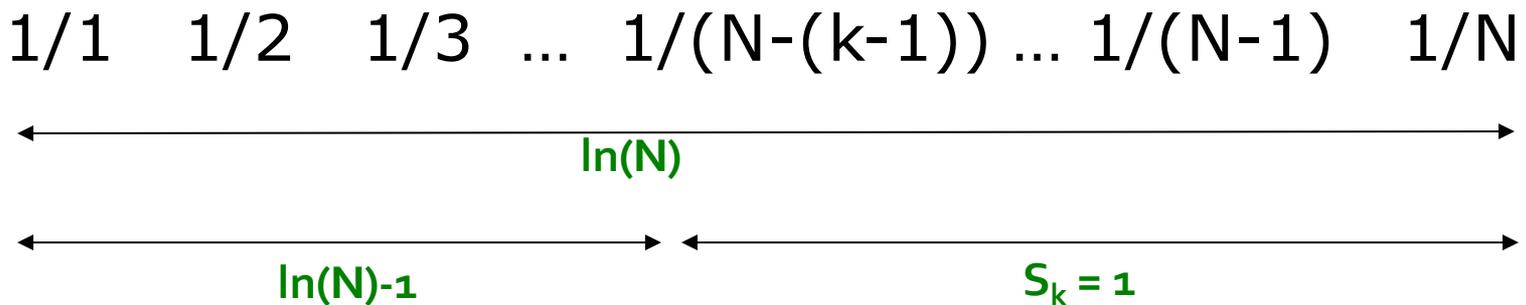 - Result due to Euler

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

ln(N)

ln(N)-1 $\qquad$ $S_k = 1$

- $S_k = 1$ implies: $H_{N-k} = ln(N) - 1 = ln(\frac{N}{e})$
- We also know: $H_{N-k} = ln(N-k)$
- **So:** $N - k = \frac{N}{e}$
- **Then:** $k = N(1 - \frac{1}{e})$

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

BALANCE: Analysis

- So after the first **k=N(1-1/e)** rounds, we cannot allocate a query to any advertiser

- **Revenue = B·N (1-1/e)**

- **Competitive ratio = 1-1/e**

- **Note: So far we assumed:**
  - **All advertisers have the same budget B**
  - **All advertisers bid 1 for the ad**
  - **(but each advertiser can bid on any subset of ads)**

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 $(1-1/e) = 0.63$**