Mining Massive Datasets

Lecture 09

Artur Andrzejak
http://pvs.ifi.uni-heidelberg.de

Note on Slides

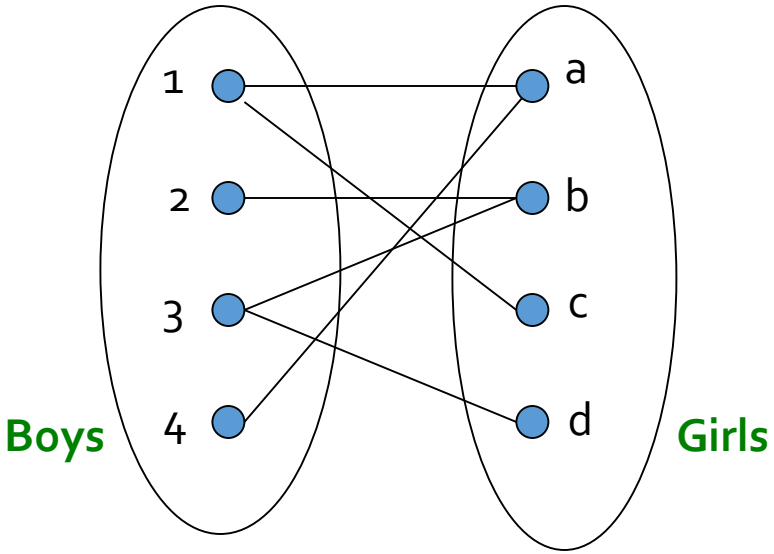
A substantial part of these slides come (either verbatim or in a modified form) from the book Mining of Massive Datasets
by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University).
For more information, see the website accompanying the book: http://www.mmds.org.

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**

Online Bipartite Matching

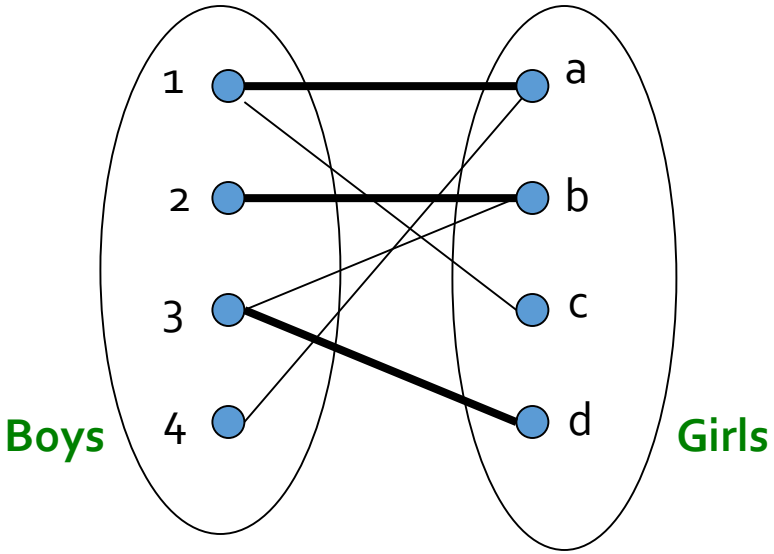
Example: Bipartite Matching



Nodes: Boys and Girls; Edges: Preferences

Goal: Match boys to girls so that maximum number of preferences is satisfied

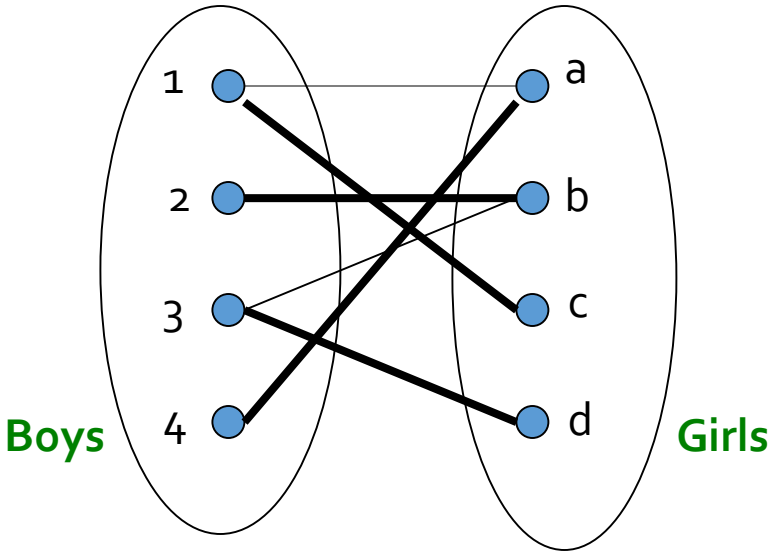
Example: Bipartite Matching



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)
- **But what if we do not know the entire graph upfront?**

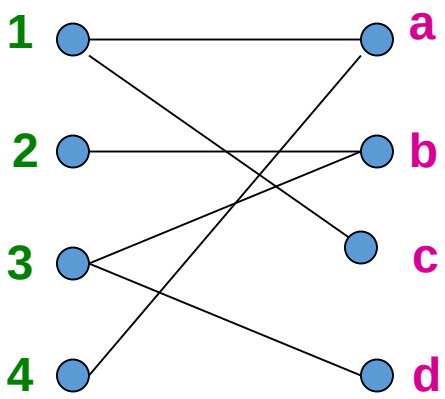
Online Graph Matching Problem

- Initially, we are given the set **boys**
- In each **round**, **one girl's choices are revealed**
  - That is, 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



(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 girl
- **How good is the algorithm?**

Competitive Ratio

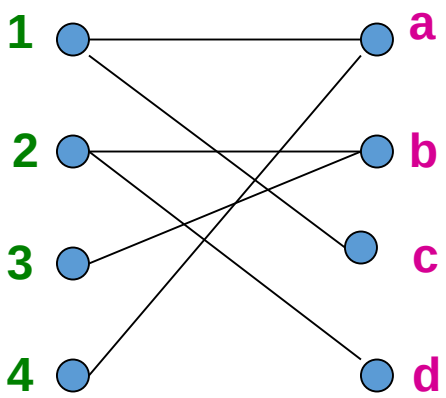
- For input I, suppose greedy produces matching M_{greedy} while an optimal matching is M_{opt}

Competitive ratio =

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

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

Worst-case Scenario

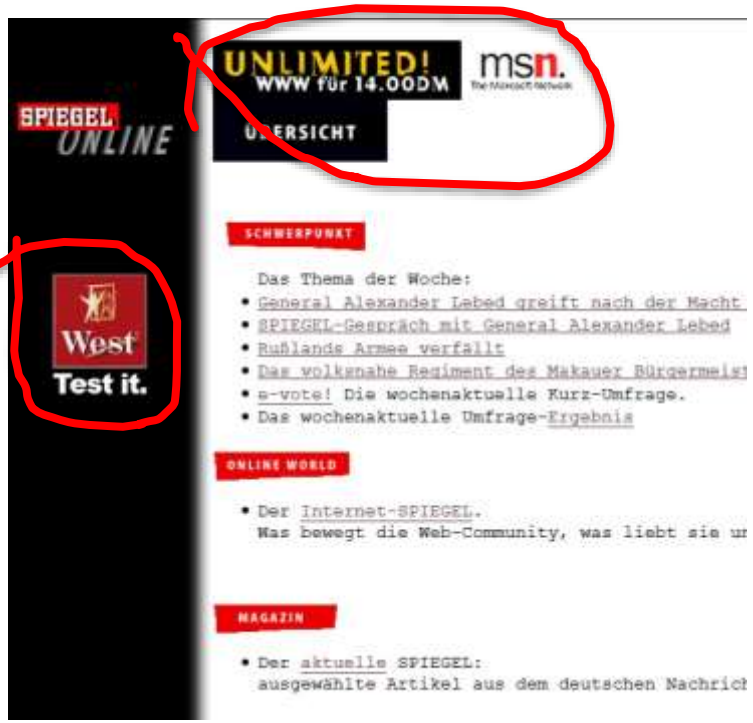


(1,a)
(2,b)

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 return of investment** for advertisers



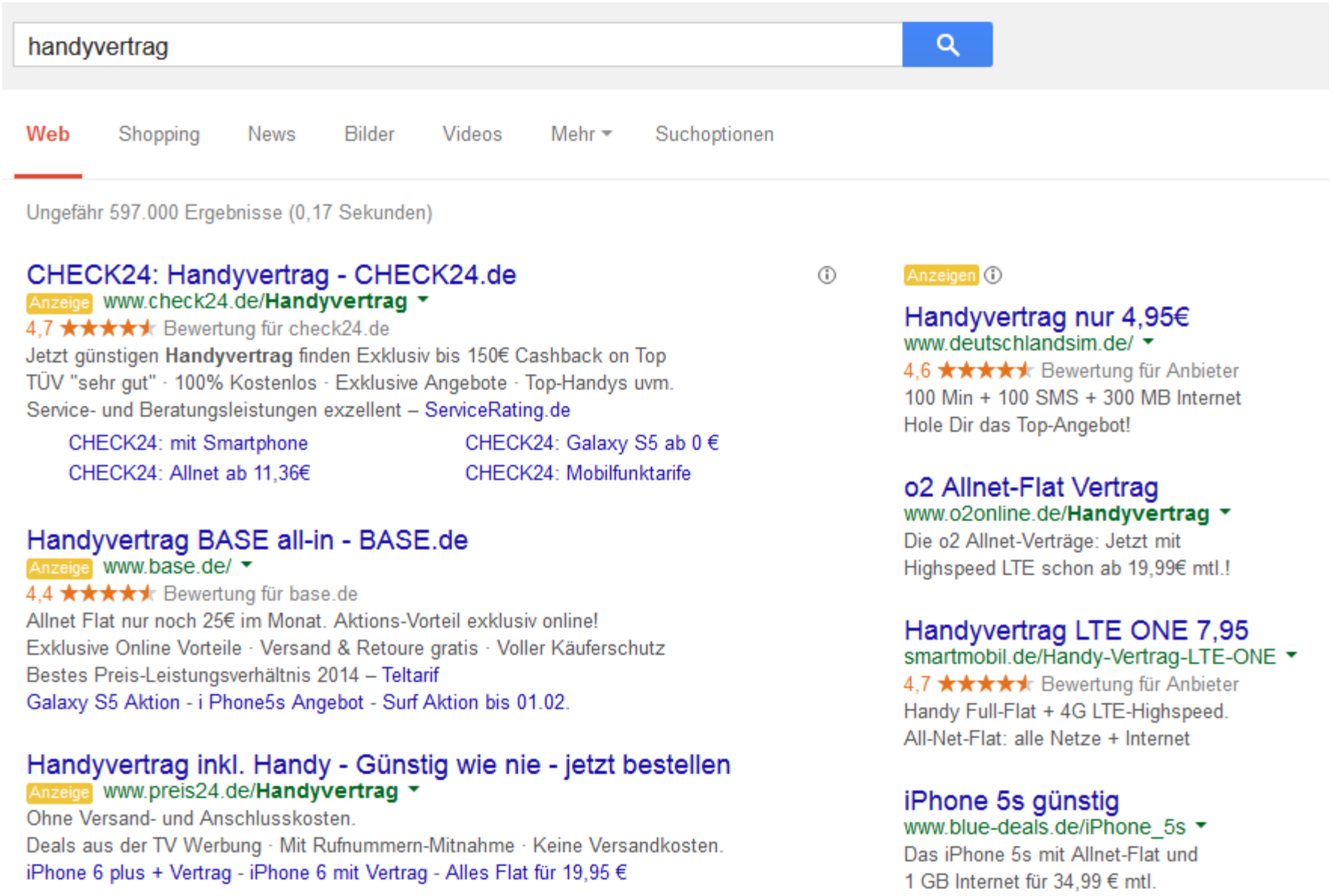
http://www.spiegel.de/static/spon1996/uber/index.html

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:**
  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: $q_1, q_2, \ldots$
- 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., **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%	1.125 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%	1.125 cents
A	$1.00	1%	1 cent

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

Complications: Budget

- **Two complications:**
  - **Budget**
  - **CTR of an ad is unknown**
- **Each advertiser has a limited budget**
  - **Search engine guarantees that the advertiser will not be charged more than their daily budget**

Complications: CTR

- **CTR: Each ad has a different likelihood of being clicked**
  - **Advertiser 1** bids $2, click probability = 0.1
  - **Advertiser 2** bids $1, click probability = 0.5
  - **Clickthrough rate (CTR)** is measured **historically**
    - **Very hard problem: 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

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
  - 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
- Algorithm:
  - For each query, assign it to an advertiser with the largest unspent budget (i.e. largest BALANCE).
    - Break ties arbitrarily (**but in a deterministic way**)

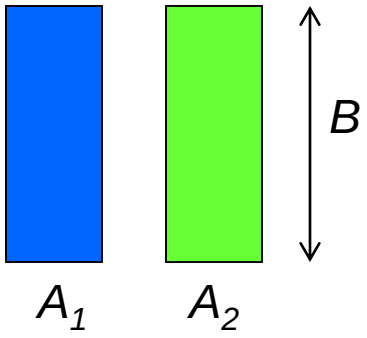
Example: BALANCE

- **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, $\mathbf{A_1}$ and $\mathbf{A_2}$, each with budget **B** (≥ 1)
  - Optimal solution exhausts both advertisers' budgets
- **BALANCE must exhaust at least one advertiser's 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 $\boldsymbol{A_2}$'s budget, but allocates $\boldsymbol{x}$ queries fewer than the optimal
  - **Revenue: $\boldsymbol{BAL = 2B - x}$**

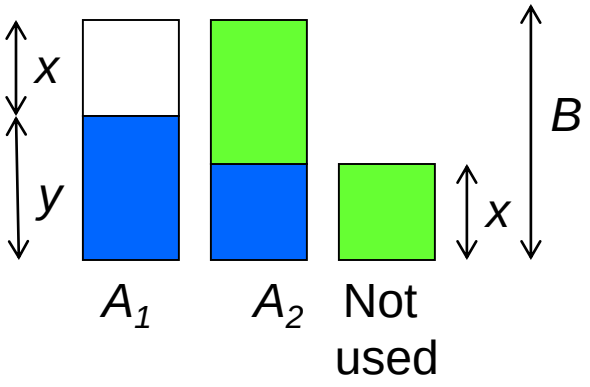
Analyzing Balance



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**



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 ≥ x**

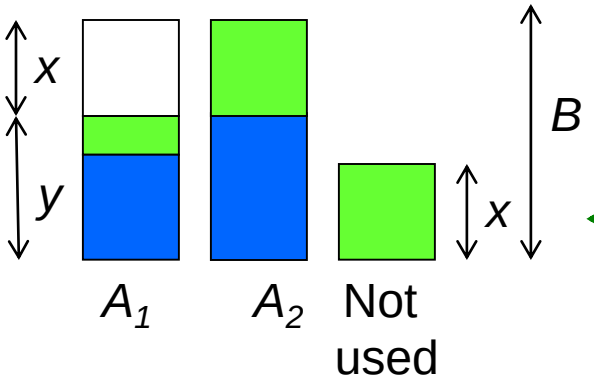
Case 1) ≤ ½ of A_1's queries got assigned to A_2
then $y \geq B/2$

Case 2) > ½ 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 exhausts A_2's budget

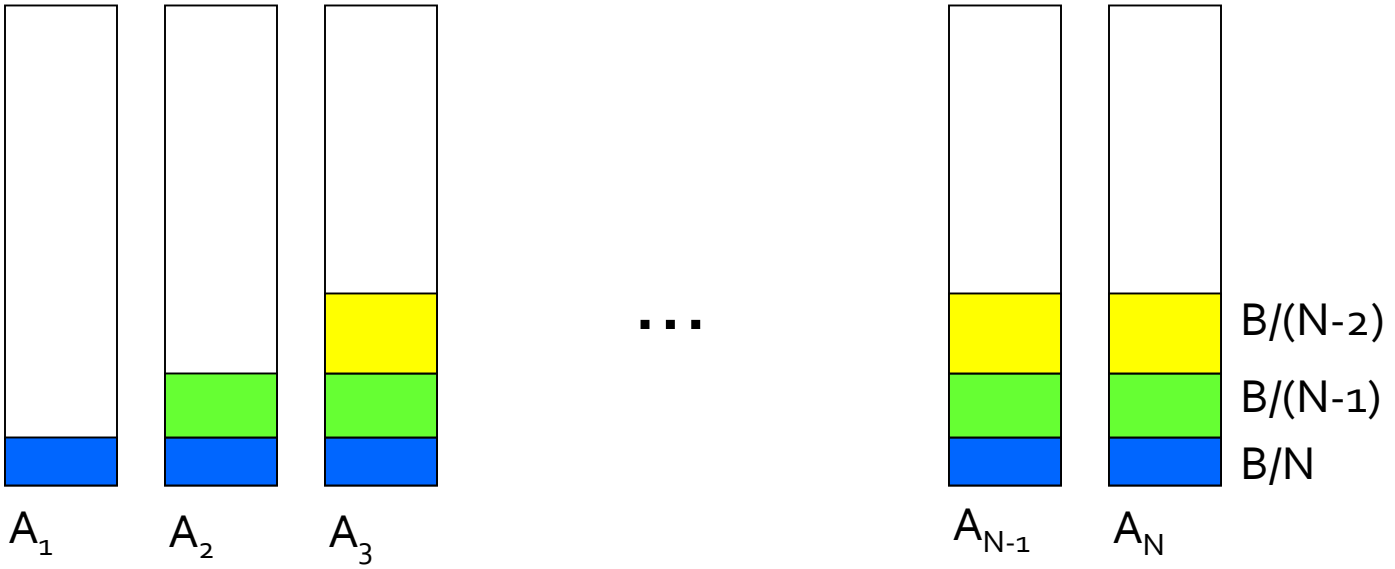
BALANCE: General Result

- **In the general case, worst competitive ratio of BALANCE is 1–1/e = approx. 0.63**
  - 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: $A_1, A_2, \ldots A_N$**
  - Each with budget $B > N$
- **Queries:**
  - $N \cdot B$ queries appear in N rounds of B queries each
- **Bidding:**
  - Round **1** queries: bidders $A_1, A_2, \ldots, A_N$
  - Round **2** queries: bidders $A_2, A_3, \ldots, A_N$
  - Round ***i*** queries: bidders $A_i, \ldots, A_N$
- **Optimum allocation:**
  Allocate round ***i*** queries to A_i
  - Optimum revenue $N \cdot B$

BALANCE Allocation



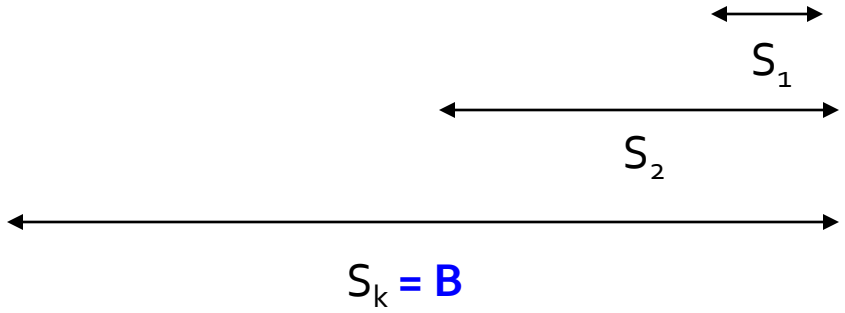
BALANCE assigns each of the queries in round 1 to **N** advertisers. After ***k*** rounds, sum of allocations to each of advertisers $A_k,\ldots,A_N$ is

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

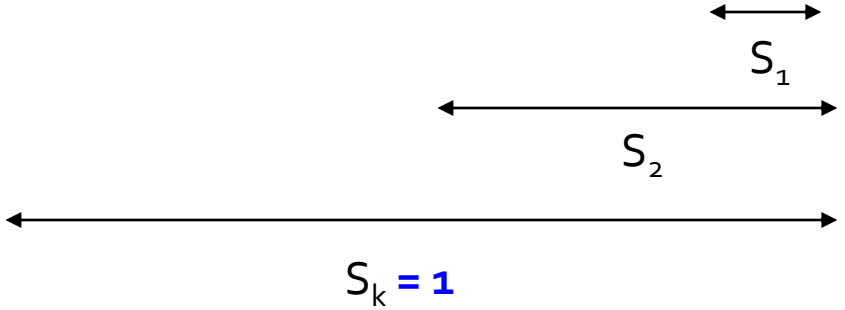
If we find the smallest *k* such that $S_k \geq B$, then after *k* rounds we cannot allocate any queries to any advertiser

BALANCE: Analysis

B/1 B/2 B/3 … B/(N-(k-1)) … B/(N-1) B/N

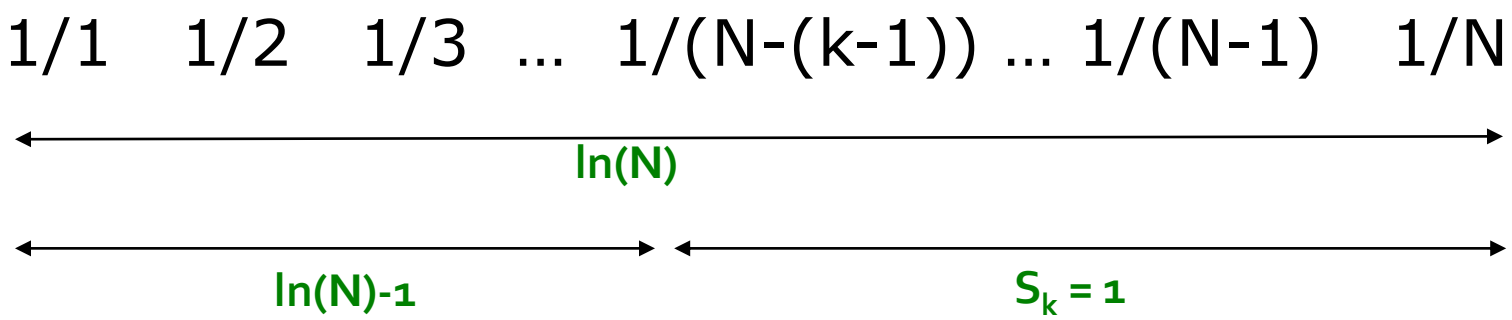


1/1 1/2 1/3 … 1/(N-(k-1)) … 1/(N-1) 1/N



BALANCE: Analysis

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



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

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**

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: bid = x_1 = **1**, b_1 = **110**
  - A_2: bid = 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 $\psi_i(q) = x_i(1-e^{-f_i})$
- Generalized Algorithm: Allocate query q to bidder i with largest value of $\psi_i(q)$
- **Same competitive ratio (1-1/e)**

Thank you.

Questions?