# Web Advertising

Thanks for source slides and material to: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

# Chapter 8 Overview

- Ability of all sorts of Web applications to support themselves through advertising
- **Most lucrative venue for on-line advertising: SEARCH**
- **Adwords model (Google): matching search queries to advertisements**
  - Algorithms for optimizing this assignment
  - **Greedy** algorithms
  - **Online** algorithms
- Selecting items to advertise at an on-line store
  - Use collaborative filtering.

# Types of Web Ads

- **Advertisers post ads directly**
  - Craig's List, auto trading sites
- **Advertisers pay for display ads to be placed on Web sites**
  - Fixed price per ***impression*** (one display of the ad with download of page by a user)
- **Online stores show ads**
  - Amazon, Macy's, etc.
  - Not paid for by manufacturers of product advertised
  - **Selected by store to maximize probability customer will buy product**
  - **Collaborative Filtering**
- **Search ads are placed among results of a search query**
  - Advertisers bid for right to have their ad shown in response to certain queries
  - **Pay only if ad is clicked on.**

# 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
  - **Make decisions without knowing the future**
  - **For search: only know past queries and current query; don’t know what queries will come in later**
  - Similar to handling data streams,
- **An online algorithm cannot always do as well as an offline algorithm.**

# Example 8.1

- **Knowing the future could help**
- **Manufacturer A** of antique furniture **bids 10 cents** on search term **"chesterfield"**
- **Manufacturer B** of conventional furniture **bids 20 cents** on both terms **"sofa" and "chesterfield"**
- Both have monthly **budget of $100**
  - **A** can place its ad **1000 times**, **B** can place its ad **500 times**
- Query "chesterfield" arrives
- Can only display one ad
- Might display B's ad because B bid more
- However, if there are many queries for "sofa" and few for "chesterfield," A will never spend its full budget
- Sending "chesterfield" queries to A might increase overall revenue
- **Without knowing the future, on-line algorithm may not perform as well as offline.**

# Greedy Algorithms

- **Make their decision in response to each input element by maximizing some function of (input element, past)**

- Example 8.2: Greedy algorithm would **assign the query to the highest bidder who still has budget left**
- Recall: **A** bid 10 cents on "chesterfield", **B** bid 20 cents on "sofa", chesterfield"

## Worst case:

- 500 "chesterfield" queries arrive
- **Greedy algorithm: all are assigned to B with the higher bid**
- B budget used up; **earns $100 revenue**
- Followed by 500 "sofa" queries that will get no ad assigned: **no revenue**

## Optimal:

- First 500 chesterfield queries assigned to A, **earn $50**
- Next 500 sofa queries assigned to B, **earn $100**
- **Total revenue $150.**

# Online Bipartite Matching

# The Matching Problem

- Simplified version of the problem of matching ads to search queries
- "Maximal matching": involves bipartite graphs with two sets of nodes
- All edges connect node on left set to node in right set.

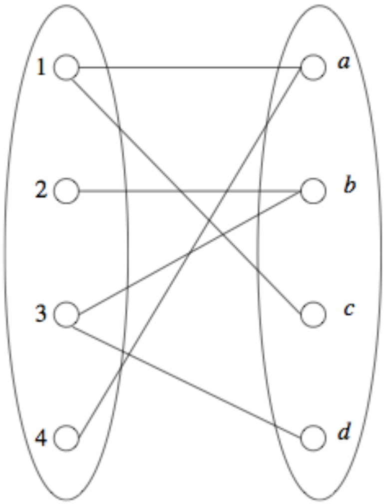
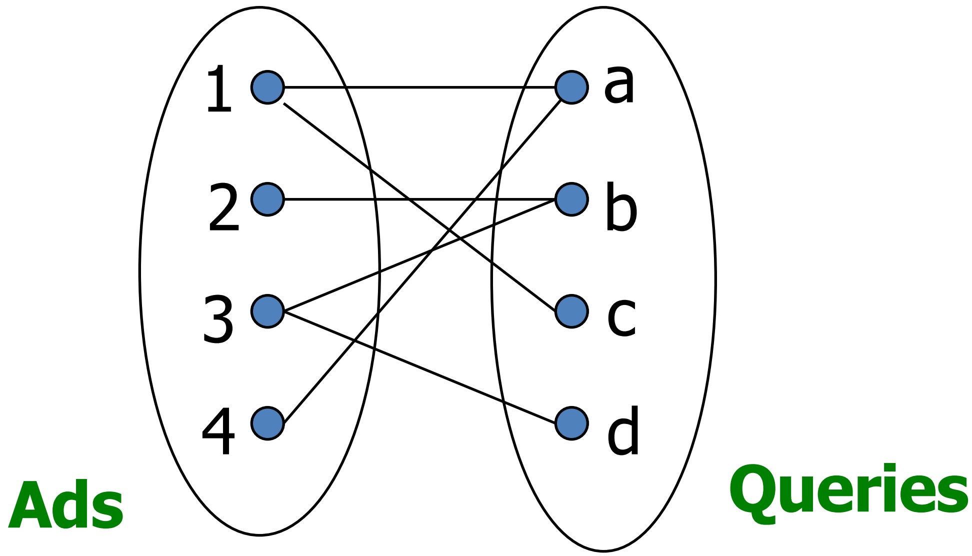



Figure 8.1: A bipartite graph

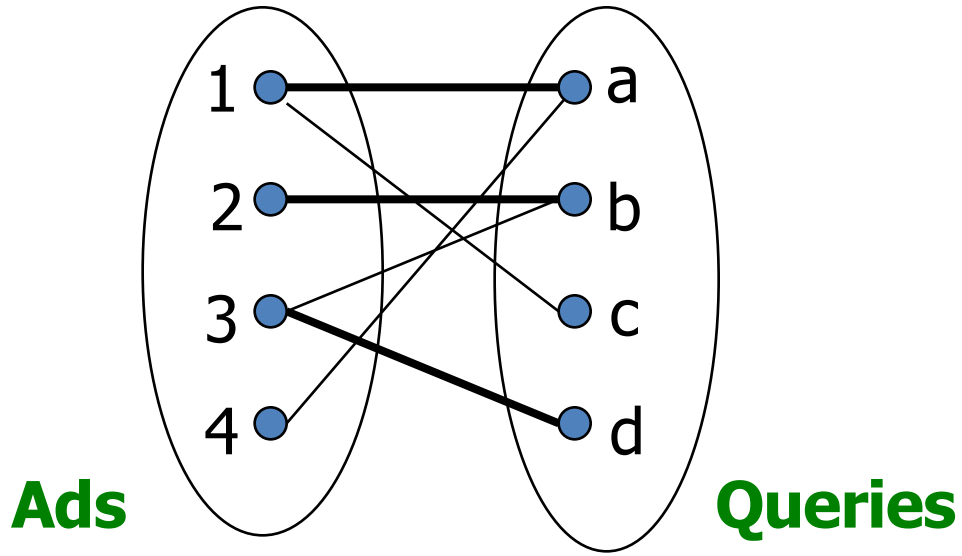
# Example: Bipartite Matching




**Nodes: Queries and Ads**

**Goal: Match queries to ads so that maximum number of matchings are made**

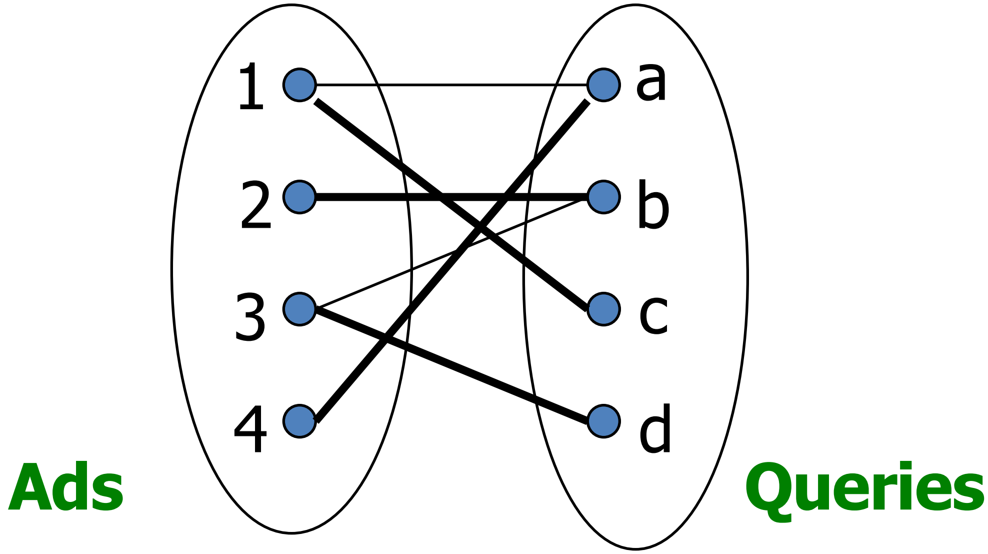
# Example: Bipartite Matching




**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
**Maximal matching**: a matching that contains the largest possible number of matches

# Matching Algorithm

- **Problem: Find a maximal matching for a given bipartite graph**
  - A perfect one if it exists

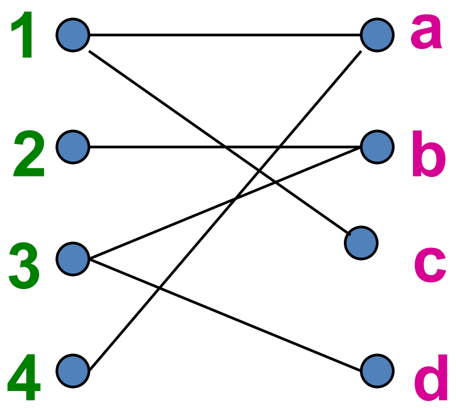
- There is a polynomial-time offline algorithm based on augmenting paths (Hopcroft & Karp 1973, see http://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm)

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

# Online Graph Matching Problem

- Initially, we are given the set **ads**
- In each **round**, **one set of query terms is added**
  - Relevant **edges** are revealed
  - Indicate which advertisers have bid on those query terms
- **At that time, we have to decide to either:**
  - Pair the **query** with an **ad**
  - Do not pair the **query** with any **ad.**

# Online Graph Matching: Example




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

# Greedy Algorithm

- **Greedy algorithm for the online graph matching problem:**
  - Pair the new query with **any** eligible ad
    - If there is none, do not pair query

- **How good is the algorithm?**

# Competitive Ratio

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

**Competitive ratio =**

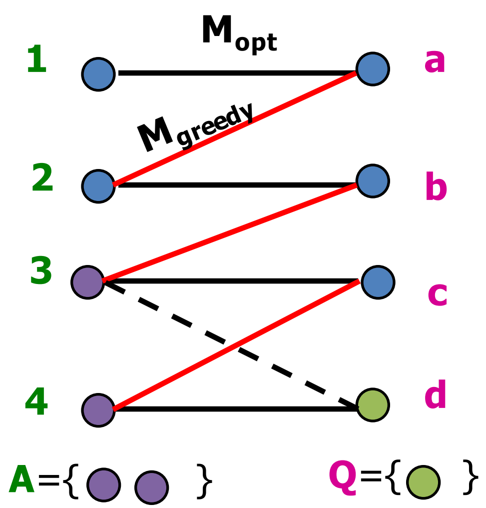
$$\min_{\text{all possible inputs } I} (|M_{greedy}|/|M_{opt}|)$$

**(what is greedy's <u>worst</u> performance <u>over all possible</u> inputs *I*).**

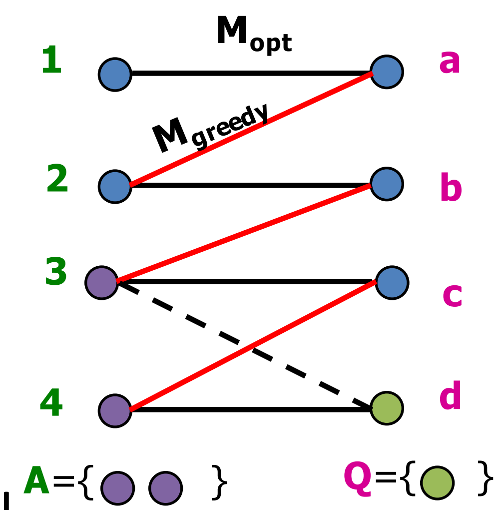
# Analyzing the Greedy Algorithm

- Consider a case: $M_{greedy} \neq M_{opt}$
- Consider the set $Q$ of queries matched in $M_{opt}$ but not in $M_{greedy}$
- **A** is the set of ads that are <u>adjacent</u> to a non-matched query in $Q$ that are already matched in $M_{greedy}$
  - Why: If there exists such a non-matched (by $M_{greedy}$) ad adjacent to a non-matched query, then greedy would have matched them
- Since ads $A$ are already matched in $M_{greedy}$ then

**(1)** $|M_{greedy}| \geq |A|$

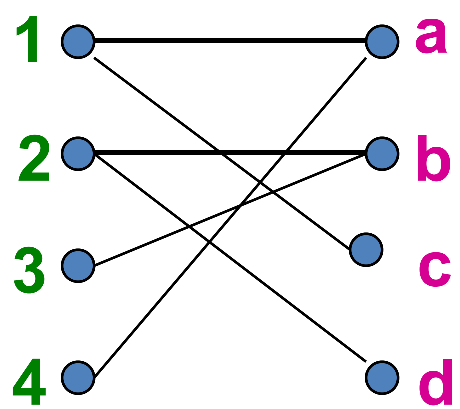


# Analyzing the Greedy Algorithm




- **Summary so far:**
  - Queries $Q$ matched in $M_{opt}$ but not in $M_{greedy}$
  - (1) $|M_{greedy}| \geq |A|$
- There are at least $|Q|$ such ads in $A$ ($|Q| \leq |A|$) otherwise the optimal algorithm couldn't have matched all queries in $Q$
  - **So:** (2) $|Q| \leq |A| \leq |M_{greedy}|$
- By definition of $Q$ also: (3) $|M_{opt}| \leq |M_{greedy}| + |Q|$
  - Worst case is when $|Q| = |A| = |M_{greedy}|$
- Combining (2) and (3)
- $|M_{opt}| \leq 2|M_{greedy}|$ **then** $|M_{greedy}|/|M_{opt}| \geq ½$
- **Competitive Ratio = ½**
- **Greedy's worst performance over all possible inputs *I*.**

# Worst-case Scenario




(1,a)
(2,b)

- **Worst case** is when $|Q| = |A| = |M_{greedy}|$
- Q = {c,d} – queries with no matching ad
- A = {1,2} – ads that are adjacent to a query in Q but are already matched to another query
- $|M_{greedy}| = 2$, $|Q| = 2$, $|A| = 2$
- **Optimal matching: (1,c), (2,d), (3,b), (4,a)**
- $|M_{opt}| = 4$
- $|M_{greedy}| / |M_{opt}| = 1/2$ **(competitive ratio)**

# Summary

- Web Advertising
- Matching search queries to advertisements
- Matching Algorithm
  - Algorithms for optimizing this assignment
  - Greedy algorithms
  - Online algorithms
- competitive ratio
  - Competitive ratio for Greedy algorithms =1/2