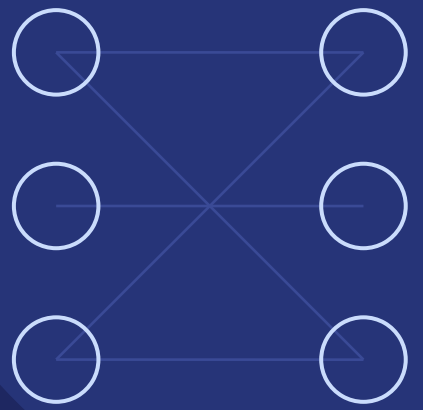
ADVANCED ALGORITHMS · FALL SEMESTER

Lecture 1

Graph Algorithms: Matchings



Faculty: K.R. Chowdhary

Professor of CS

Bipartite matching, Hall's theorem, and the augmenting-path algorithm

The Matching Problem

Assigning teachers to courses

- Each teacher is qualified to teach certain courses, but not others.
- We assign each course to a teacher so that no two teachers get the same course.
- When full assignment is impossible, we assign as many teachers as possible and stop.

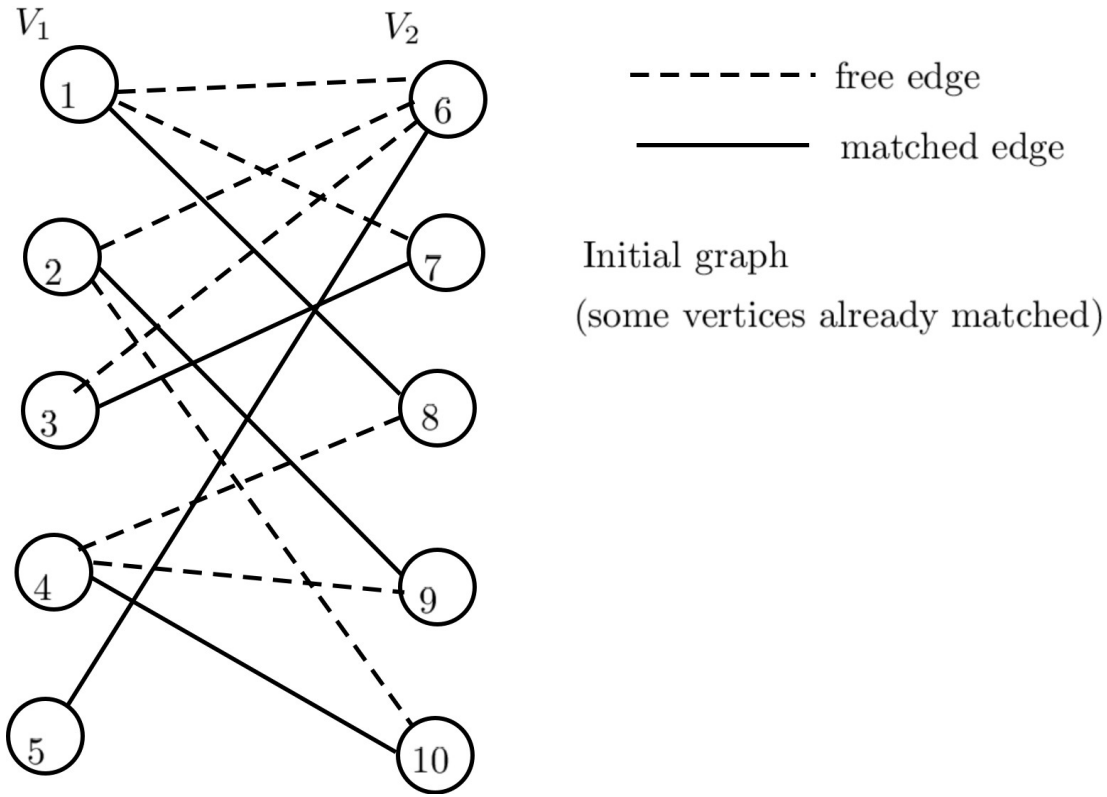


Figure: Bipartite graph — teachers V_1, courses V_2

Matching, Maximal, and Complete Matching

Definition 1.1 — Matching

Given a graph $G = (V, E)$, a matching M is a subset of the edges such that no two edges in M share a common vertex. The cardinality of M is the size of the matching.

Maximal matching

A matching of maximum possible size — no larger matching exists in the graph.

Complete matching

Every vertex is an endpoint of some edge in the matching. Every complete matching is also maximal.

Exact vs. inexact

Exact matching requires strict correspondence; inexact matching allows the compared graphs to differ structurally.

Hall's Marriage Theorem

Theorem 1.2 — Hall's theorem: necessary and sufficient conditions for a matching that entirely covers X.

Let G be a bipartite graph with parts X and Y, so G = (X, Y, E). For a set W of vertices in X, let $N_a(W)$ denote its neighborhood — all vertices in Y adjacent to some element of W.

There is a matching that entirely covers X if and only if, for every subset W of X:

$$|W| \leq |N_a(W)|$$

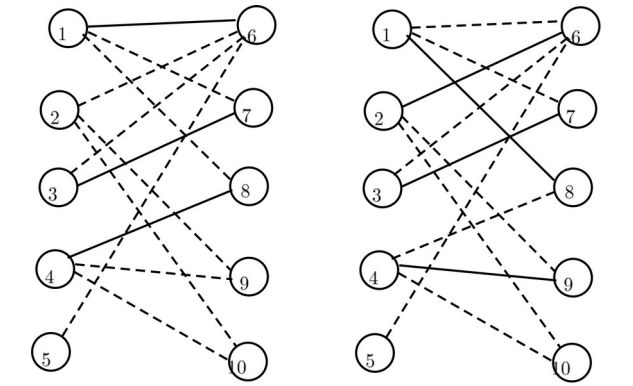
In words: every subset W of X must have sufficiently many adjacent vertices in Y. The theorem confirms when a perfect matching exists, but doesn't by itself give an algorithm to find one.

Augmenting Paths

Definition 1.3 — Augmenting path

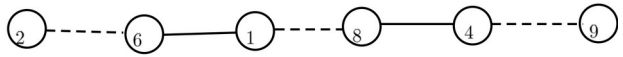
A path connecting two unmatched vertices in which alternate edges are in M. It must have odd length, and begin and end with edges not in M.

Given augmenting path P, we get a bigger matching M ⊕ P by removing from M the edges that are in P, then adding the edges of P not initially in M (⊕ is exclusive-OR on edge sets).



(a) Matching (M)

(c) Larger mtching ($M \oplus P$).



(b) Augmented path (P)

CORE TECHNIQUE

The Maximal-Matching Algorithm

Theorem 1.5: *A matching in G is maximum if and only if there is no augmenting path in G with respect to M.*

ALGORITHM — MAXIMAL MATCHING FOR G = (V, E)

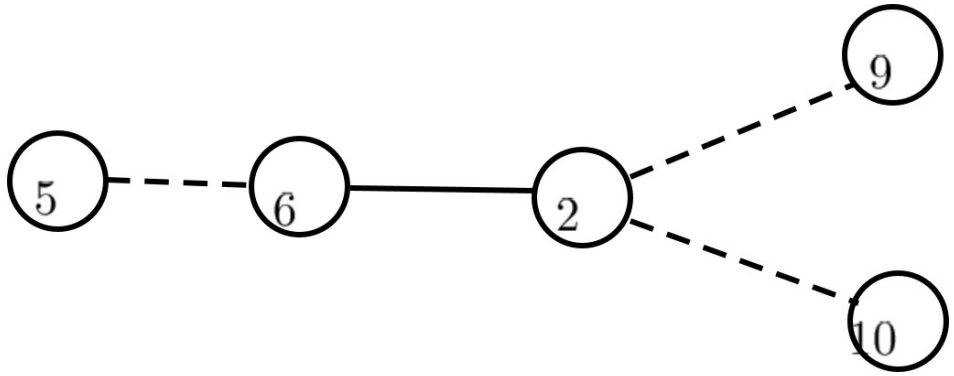
1. Start with M = ∅.
2. Find an augmenting path P relative to M, and replace M by M ⊕ P.
3. Repeat step 2 until no augmenting path exists — M is now a maximal matching.

Remaining question: how do we actually find an augmenting path relative to M? — next slide.

Finding an Augmenting Path (Bipartite Case)

Build a level graph, BFS-style

- Level 0: all unmatched vertices from V_1.
- Odd level i: add vertices adjacent to level i−1 via a non-matching edge (plus that edge).
- Even level i: add vertices adjacent to level i−1 via a matching edge (plus that edge).
- Stop when an unmatched vertex is added at an odd level, or no more vertices can be added.



Vertex 5 unmatched at level 0. Level 1 adds edge (5,6); level 2 adds matching edge (6,2); level 3 adds (2,9) or (2,10) — both unmatched, giving two valid augmenting paths.

ANALYSIS

Running Time of the Algorithm

Let graph G have n vertices and e edges (adjacency-list representation).

O(e)

to build one augmenting-path graph for a given matching

n/2

augmenting paths built at most — each grows the matching by ≥ 1 edge

O(ne)

total time to find a maximal matching in a bipartite graph

Where Matching Shows Up

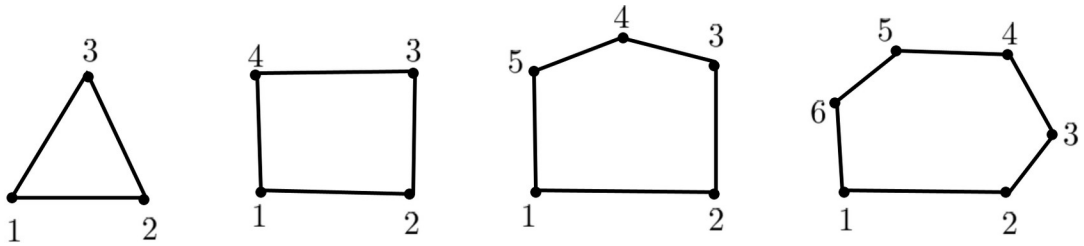
Matchings are the basis of many optimization problems — from assigning workers to jobs, to scheduling jobs with precedence constraints across processors, to structural chemistry and multi-sensor object localization.

1. Finding correspondence between elements of data schemes or data instances.
2. Consolidating related databases from merging organizations, matching their schemas.
3. Matching in UML class taxonomies, ER diagrams, and ontologies.
4. Relational database matching (SQL-driven) vs. hierarchical-relationship matching — the approach differs by domain.
5. Graph matching for structured data: finding correspondence between nodes/edges of two graphs so that substructures align.

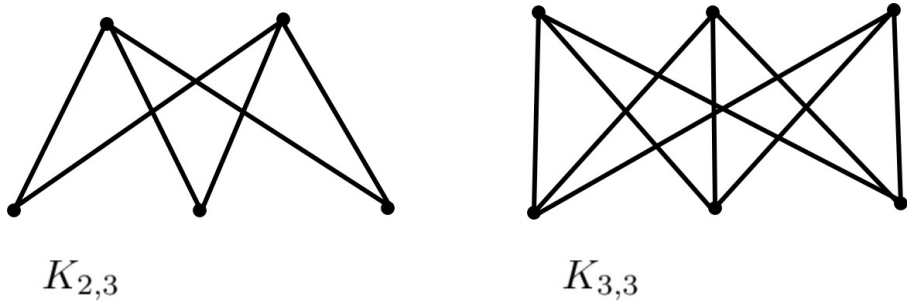
PRACTICE

Exercises

- Find all connected components of a graph.
- Determine whether a graph of n vertices has a cycle.
- Find all simple cycles of a graph — how many can there be, and what's the time complexity?
- Which graphs below are bipartite?
- Identify the $K_{m,n}$ graphs below — what's the edge-count formula?
- Max edges in a maximum matching of a bipartite graph with n vertices?
- Define: maximal matching, maximum matching, perfect matching.
- Determine the complexity of the maximal-matching algorithm from class.
- Rewrite the algorithm to find a minimum (not maximum) matching.



Are these bipartite?



$K_{m,n}$ graphs

References

[1] **Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman,**
"Data Structures and Algorithms" — Pearson Education, India, 2002

[2] **Allen B. Tucker, Jr.,**
"The Computer Science and Engineering Handbook" — CRC Press, 1997

Disclaimer: these notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the instructor.