

# Algorithms

*Definition:* An **algorithm** is a set of rules followed to solve a problem.

In general, an algorithm has the steps:          Havel–Hakimi:

- 1 Organize the input.
- 2 Repeatedly apply some steps until a termination condition holds
- 3 Analyze data upon termination

Computers can be used to run the algorithms once we verify they work.

To verify the **correctness** of an algorithm:

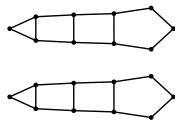
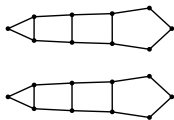
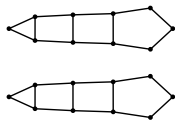
- 1 Verify that the algorithm terminates. (often invoking finiteness)
- 2 Verify that the result satisfies the desired conditions.

## Matchings in Graphs

*Definition:* A **matching**  $M$  in a graph  $G$  is a subset of edges of  $G$  that share no vertices.

*Definition:* A **maximal matching**  $M$  is a matching such that the inclusion into  $M$  of any edge of  $G \setminus M$  is no longer a matching.

*Definition:* A **maximum matching** is a matching  $M$  that has the most edges possible for the graph  $G$ .



*Thought Exercise:* What is the result of overlapping two matchings?

*Recall.* A **perfect matching** is a matching involving every vertex of  $G$ .

★ We will discuss matchings in a bipartite graph ★

## Application: Scheduling

Suppose you are working in a group trying to complete all the problems on the homework. Depending on everyone's preferences, you would like to assign each member one problem to do.

Person A likes problems 9-1, 9-2, 9-3, and 9-5.

Person B likes problems 9-1, 9-2, and 9-4.

Person C likes problems 9-3, 9-4, and 9-5.

Person D likes problems 9-2 and 9-3.

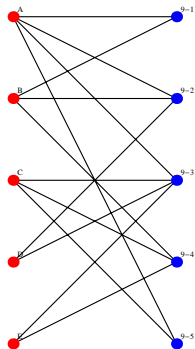
Person E likes problems 9-3 and 9-4.

Create a graph that models the situation.

*Question:*

What is a maximum matching for this graph?

We will use an algorithm to answer this question.



## Motivating The Hungarian Algorithm

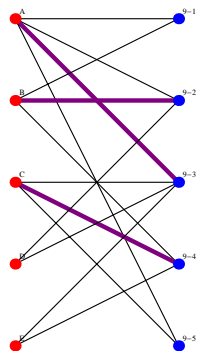
Let us work through the basic idea behind the algorithm. We start with an initial matching; we might as well make it maximal. Why is the pictured matching maximal?

**Definition:** Given a matching  $M$  in a graph  $G$ , an  $M$ -**alternating path** is a path in  $G$  that starts at a vertex not in  $M$ , and whose edges alternate between being in  $M$  and not in  $M$ .

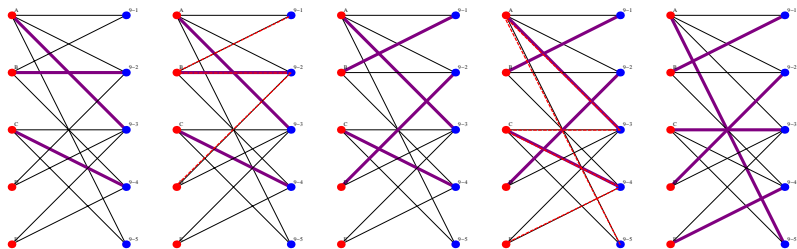
**Example**  $D \rightarrow 9-2 \rightarrow B \rightarrow 9-4 \rightarrow C$  is an  $M$ -alternating path.

**Definition:** An  $M$ -**augmenting path** is an  $M$ -alternating path that begins AND ends at unmatched vertices.

It is **augmenting** because we can improve  $M$  by toggling the edges between those in  $M$  and those not in  $M$ .



# Motivating The Hungarian Algorithm



Given  $M$ ,  $P = D \rightarrow 9-2 \rightarrow B \rightarrow 9-1$  is an  $M$ -augmenting path. Toggling the edges in  $P$  gives a new matching  $M'$ .

Given  $M'$ ,  $P' = E \rightarrow 9-4 \rightarrow C \rightarrow 9-3 \rightarrow A \rightarrow 9-5$  is an  $M'$ -augmenting path. Toggling the edges in  $P'$  gives a new matching  $M''$ .

The matching  $M''$  is maximal.

# The Hungarian Algorithm

*The Hungarian Algorithm* (Kuhn, König, Egeváry) [*Finds a maximum matching in a bipartite graph (w/red and blue vertices)*]

- 1 Start with a bipartite graph  $G$  and any matching  $M$ .  
Label all red vertices *eligible* (for augmentation).
- 2 If all red, eligible vertices are matched, stop. Otherwise, there exists a red, unmatched, eligible vertex to use in the next step.
- 3 Let  $v$  be an unmatched, eligible, red vertex. Start growing all possible  $M$ -alternating paths from  $v$ . That is, follow every edge not in  $M$  to a blue vertex. From a matched blue vertex, follow the edge of  $M$  back to a red vertex, and repeat as far as possible.

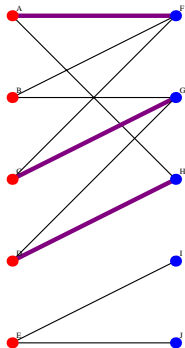
{ If there is an  $M$ -augmenting path, toggle edges to augment  $M$ .  
{ If there is no  $M$ -augmenting path, mark  $a$  ineligible.

Return to Step 2.

## Applying the Hungarian Algorithm

Here is something that might happen during an application of the Hungarian algorithm:

*Example.* There is no  $M$ -augmenting path starting at  $B$  in the graph to the right.



We would mark  $B$  ineligible and move on to the next eligible, unmatched red vertex in the graph ( $E$ ).

## Proof of Correctness

*Claim.* The Hungarian Algorithm gives a maximum matching.

*Proof.* We must show that the algorithm always stops, and that when it stops, the output is indeed a maximum matching.

**The algorithm terminates.** Each time Step 3 is run, one red vertex either becomes matched or becomes ineligible. Also, no red vertex that starts matched becomes unmatched. Since there are a finite number of red vertices, the algorithm must terminate.

**The output is a maximum matching.** The output  $M$  is a matching inducing no  $M$ -augmenting paths in the graph. Suppose that there were another matching  $M^*$  that used more edges than  $M$ .

When we overlap  $M$  and  $M^*$ , the result is a union of cycles and paths. At least one path must have more edges from  $M^*$  than  $M$ .

This path is an  $M$ -augmenting path, contradicting the definition of  $M$ .