MATH 322 – Graph Theory
Fall Term 2021

Notes for Lecture 22

Tuesday, November 30

Notions related to one-factors and one-factorizations of undirected graphs

A related notion

Definition

Let $G = (V, E)$ be a graph. A subset E' of E is called a *matching* in G if, for any two different edges $e_1, e_2 \in E'$, we have that e_1, e_2 are not adjacent *(that is, they don't have any common endvertex).*

In other words, E' is a matching in G if it is the edge set of a 1-regular subgraph of G (where we consider the vertex set of the subgraph to be all the endvertices of the edges in E').

A matching E' in G is called a *perfect matching* if every vertex of G is *covered* by E', that is, if every vertex of G is the endvertex of some edge in E'. In other words, E' is a perfect matching if it is the edge set of a one-factor of G.

Definition

The maximum cardinality of a matching in G is denoted by $\nu(G)$.

A related notion

Definition

Let $G = (V, E)$ be a graph. A subset E' of E is called a *matching* in G if, for any two different edges $e_1, e_2 \in E'$, we have that e_1, e_2 are not adjacent *(that is, they don't have any common endvertex).*

In other words, E' is a matching in G if it is the edge set of a 1-regular subgraph of G (where we consider the vertex set of the subgraph to be all the endvertices of the edges in E').

A matching E' in G is called a *perfect matching* if every vertex of G is *covered* by E', that is, if every vertex of G is the endvertex of some edge in E'. In other words, E' is a perfect matching if it is the edge set of a one-factor of G.

Definition

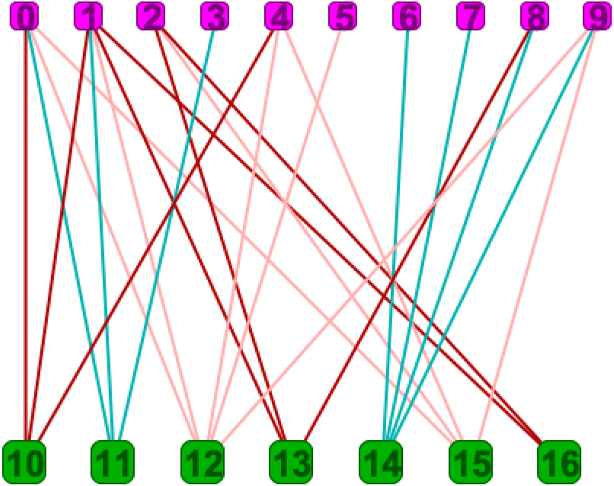
The maximum cardinality of a matching in G is denoted by $\nu(G)$.

Useful observation

Let G be a graph. Then $\nu(G)$ coincides with the independence number $\alpha(L(G))$ of the line graph $L(G)$ of G (why?).

Examples motivated by real-life

As we said last time, looking for matchings makes even more sense in bipartite graphs (here, as we will see, it's useful to discuss both complete and non-complete bipartite graphs).



Some real-life interpretations of matchings in (complete or non-complete) bipartite graphs

- Suppose one partite set of the graph represents workers in a factory, while the other partite set represents the machines they can operate (and there's an edge connecting a worker with a machine **if and only if** that worker knows how to operate that machine). Suppose also that we can't have two or more workers operating the same machine at the same time.

  We would of course want to have as many workers as possible operating (some of) the machines at the same time (ideally we would like to have all workers being able to work at the same time),

  or, **in the case that we have more workers than machines,** we would like to have as many machines as possible being in operation at the same time (ideally all the machines would be in operation).

  This corresponds to finding a matching in the graph which has maximum possible cardinality.

Some real-life interpretations of matchings in (complete or non-complete) bipartite graphs

- Suppose one partite set of the graph represents (undergraduate or graduate) positions at Canadian universities, while the other partite set represents prospective students (and there's an edge connecting a student with a university (thus in our model with all the available positions at this university) **if and only if** this student has applied to that university).

  We would like to:

  - either fill all available positions,
  - or have all students get accepted to one of the universities they applied to.

  This corresponds to finding a matching in the graph which has maximum cardinality.

Some real-life interpretations of matchings in (complete or non-complete) bipartite graphs

- Suppose one partite set of the graph represents (undergraduate or graduate) positions at Canadian universities, while the other partite set represents prospective students (and there's an edge connecting a student with a university (thus in our model with all the available positions at this university) **if and only if** this student has applied to that university).

  We would like to:

  - either fill all available positions,
  - or have all students get accepted to one of the universities they applied to.

  This corresponds to finding a matching in the graph which has maximum cardinality.

We will revisit a version of this problem which takes into account even more criteria very shortly.

Matching in bipartite graphs

Theorem 2 (Hall, 1935)

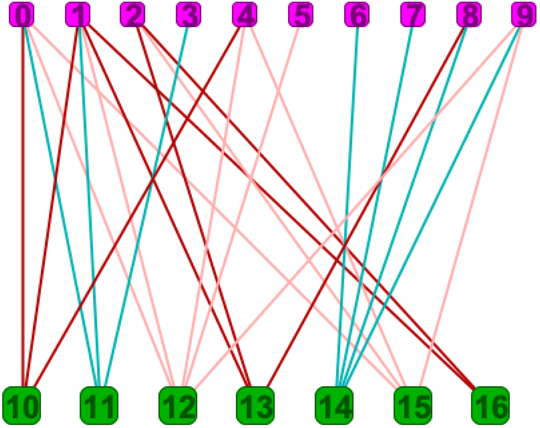
Let $G = (V, E)$ be a (not necessarily complete) bipartite graph with partite sets A and B (that is, $V(G) = A \cup B$, and there are no edges in $E(G)$ joining two vertices in A, or two vertices in B).

Then there is a matching in G covering the set A **if and only if**

$$\text{for every } S \subseteq A, \text{ we have that } |S| \leqslant |N(S)|,$$

where $N(S)$ is the union of all neighbourhoods of vertices in S.

Testing this on an example



Matching in bipartite graphs

In real-life applications, it's also very useful to have a *defect version of Hall's theorem.*

Proposition 3 (Corollary to Hall's theorem)

Let $G = (V, E)$ be a (not necessarily complete) bipartite graph with partite sets A and B.

Assume that, for some integer $d \geqslant 1$, G satisfies the following:

for every $S \subseteq A$, we have that $|N(S)| \geqslant |S| - d$.

Then there is a matching in G of cardinality at least $|A| - d$ (in other words, a matching that covers at least $|A| - d$ of the vertices in A).

Matching in bipartite graphs

Proof of Proposition 3 (defect version of Hall's theorem). We add d new vertices to the set B, say vertices $w_1, w_2, \ldots, w_d$, and we join each of these vertices **with each vertex from A.** The new graph G' that we have constructed is again a bipartite graph with partite sets A and $B \cup \{w_1, w_2, \ldots, w_d\}$.

Matching in bipartite graphs

Proof of Proposition 3 (defect version of Hall's theorem). We add d new vertices to the set B, say vertices $w_1, w_2, \ldots, w_d$, and we join each of these vertices **with each vertex from A.** The new graph G' that we have constructed is again a bipartite graph with partite sets A and $B \cup \{w_1, w_2, \ldots, w_d\}$.

We now check that G' satisfies the condition of Hall's theorem: for every non-empty $S \subseteq A$, we have that $N_{G'}(S)$, that is, the neighbourhood of the set S in the graph G', contains all the vertices from G that are neighbours with vertices from S in the original graph G, as well as all the new vertices (since each of these new vertices is adjacent to every vertex in A).

In other words,

$$N_{G'}(S) = N_G(S) \cup \{w_1, w_2, \ldots, w_d\}$$

$$\Rightarrow \quad |N_{G'}(S)| = |N_G(S)| + |\{w_1, w_2, \ldots, w_d\}| = |N_G(S)| + d \geqslant \big(|S| - d\big) + d = |S|.$$

Matching in bipartite graphs

Proof of Proposition 3 (defect version of Hall's theorem). We add d new vertices to the set B, say vertices $w_1, w_2, \ldots, w_d$, and we join each of these vertices **with each vertex from A.** The new graph G' that we have constructed is again a bipartite graph with partite sets A and $B \cup \{w_1, w_2, \ldots, w_d\}$.

We now check that G' satisfies the condition of Hall's theorem: for every non-empty $S \subseteq A$, we have that $N_{G'}(S)$, that is, the neighbourhood of the set S in the graph G', contains all the vertices from G that are neighbours with vertices from S in the original graph G, as well as all the new vertices (since each of these new vertices is adjacent to every vertex in A).

In other words,

$$N_{G'}(S) = N_G(S) \cup \{w_1, w_2, \ldots, w_d\}$$

$$\Rightarrow \quad |N_{G'}(S)| = |N_G(S)| + |\{w_1, w_2, \ldots, w_d\}| = |N_G(S)| + d \geqslant \big(|S| - d\big) + d = |S|.$$

Thus, by Hall's theorem the new graph G' has a matching M_0' which covers A.

We now observe that M_0' can contain **at most** d edges with one endvertex from the new vertex subset $\{w_1, w_2, \ldots, w_d\}$ that we introduced. Thus, if we remove these edges from M_0', we get a matching M_0 of the original graph G which will have cardinality $\geqslant |M_0'| - d = |A| - d$.

Stable matchings?

Important Remark. The only (complete) bipartite graphs for which there exists a perfect matching are the graphs $K_{n,n}$ (where $n \geqslant 1$ is some integer).

In fact, in the graph $K_{n,n}$ we will have $n!$ different perfect matchings (or equivalently $n!$ different one-factors). *(Can you explain why this is so?)*

Stable matchings?

Important Remark. The only (complete) bipartite graphs for which there exists a perfect matching are the graphs $K_{n,n}$ (where $n \geqslant 1$ is some integer).

In fact, in the graph $K_{n,n}$ we will have $n!$ different perfect matchings (or equivalently $n!$ different one-factors). *(Can you explain why this is so?)*

Related to also ask: Can we find a one-factorisation of $K_{n,n}$ too? And if yes, how many of the abovementioned perfect matchings should we combine to get a one-factorisation? *(It's easier if you described specific matchings you would combine.)*

Stable matchings?

Important Remark. The only (complete) bipartite graphs for which there exists a perfect matching are the graphs $K_{n,n}$ (where $n \geqslant 1$ is some integer).

In fact, in the graph $K_{n,n}$ we will have $n!$ different perfect matchings (or equivalently $n!$ different one-factors). *(Can you explain why this is so?)*

Related to also ask: Can we find a one-factorisation of $K_{n,n}$ too? And if yes, how many of the abovementioned perfect matchings should we combine to get a one-factorisation? *(It's easier if you described specific matchings you would combine.)*

Answer. Let us write $\{1, 2, \ldots, n\} \cup \{1', 2', \ldots, n'\}$ for the vertex set of $K_{n,n}$ (with the two subsets here being the two partite sets of $K_{n,n}$).

Then a one-factorisation of $K_{n,n}$ can be formed by considering the following n matchings (which are pairwise disjoint edge subsets and form a partition of $E(K_{n,n})$):

$$\big\{\{1,1'\},\, \{2,2'\},\, \ldots,\, \{(n-1),(n-1)'\},\, \{n,n'\}\big\},$$

$$\big\{\{1,2'\},\, \{2,3'\},\, \ldots,\, \{(n-1),n'\},\, \{n,1'\}\big\},$$

$$\big\{\{1,3'\},\, \{2,4'\},\, \ldots,\, \{(n-1),1'\},\, \{n,2'\}\big\},$$

$$\ddots \quad \ddots \quad \ddots \quad \ddots \quad \ddots \quad \ddots \quad \ddots$$

$$\big\{\{1,n'\},\, \{2,1'\},\, \ldots,\, \{(n-1),(n-2)'\},\, \{n,(n-1)'\}\big\}.$$

Conclusions

– The only (complete) bipartite graphs for which there exists a perfect matching are the graphs $K_{n,n}$ (where $n \geqslant 1$ is some integer).

– In fact, in the graph $K_{n,n}$ we will have $n!$ different perfect matchings (or equivalently $n!$ different one-factors).

– Moreover, we can find one-factorisations too.

Given the above remarks, in such graphs we don't simply look for a matching, but we start introducing even more criteria regarding whether a certain matching is preferable over another matching.

The Stable Marriage Problem

Suppose that we denote the bipartition of $K_{n,n}$ by (M, W), that is, we write M for the first partite set of $K_{n,n}$ and W for the second one.

Suppose also that M represents men and W represents women, and we want to have each man from M to get 'engaged to be married' to a woman from W. **Here we have the additional assumption that each man from M has ranked all women from W in (descending) order of preference, and similarly each woman from W has ranked all men from M in (descending) order of preference.**

Clearly what we want is a perfect matching of $K_{n,n}$. Moreover...

The Stable Marriage Problem

Suppose that we denote the bipartition of $K_{n,n}$ by (M, W), that is, we write M for the first partite set of $K_{n,n}$ and W for the second one.

Suppose also that M represents men and W represents women, and we want to have each man from M to get 'engaged to be married' to a woman from W. **Here we have the additional assumption that each man from M has ranked all women from W in (descending) order of preference, and similarly each woman from W has ranked all men from M in (descending) order of preference.**

Clearly what we want is a perfect matching of $K_{n,n}$. Moreover...

we will consider such a matching *not stable* if it contains (at least) two pairs (m_{i_1}, w_{j_1}) and (m_{i_2}, w_{j_2}) such that

The Stable Marriage Problem

Suppose that we denote the bipartition of $K_{n,n}$ by (M, W), that is, we write M for the first partite set of $K_{n,n}$ and W for the second one.

Suppose also that M represents men and W represents women, and we want to have each man from M to get 'engaged to be married' to a woman from W. **Here we have the additional assumption that each man from M has ranked all women from W in (descending) order of preference, and similarly each woman from W has ranked all men from M in (descending) order of preference.**

Clearly what we want is a perfect matching of $K_{n,n}$. Moreover...

we will consider such a matching *not stable* if it contains (at least) two pairs (m_{i_1}, w_{j_1}) and (m_{i_2}, w_{j_2}) such that

1. m_{i_1} prefers w_{j_2} over w_{j_1},

The Stable Marriage Problem

Suppose that we denote the bipartition of $K_{n,n}$ by (M, W), that is, we write M for the first partite set of $K_{n,n}$ and W for the second one.

Suppose also that M represents men and W represents women, and we want to have each man from M to get 'engaged to be married' to a woman from W. **Here we have the additional assumption that each man from M has ranked all women from W in (descending) order of preference, and similarly each woman from W has ranked all men from M in (descending) order of preference.**

Clearly what we want is a perfect matching of $K_{n,n}$. Moreover...

we will consider such a matching *not stable* if it contains (at least) two pairs (m_{i_1}, w_{j_1}) and (m_{i_2}, w_{j_2}) such that

1. m_{i_1} prefers w_{j_2} over w_{j_1},
2. and w_{j_2} also prefers m_{i_1} over m_{i_2}.

The Stable Marriage Problem

Suppose that we denote the bipartition of $K_{n,n}$ by (M, W), that is, we write M for the first partite set of $K_{n,n}$ and W for the second one.

Suppose also that M represents men and W represents women, and we want to have each man from M to get 'engaged to be married' to a woman from W. **Here we have the additional assumption that each man from M has ranked all women from W in (descending) order of preference, and similarly each woman from W has ranked all men from M in (descending) order of preference.**

Clearly what we want is a perfect matching of $K_{n,n}$. Moreover...

we will consider such a matching *not stable* if it contains (at least) two pairs (m_{i_1}, w_{j_1}) and (m_{i_2}, w_{j_2}) such that

1. m_{i_1} prefers w_{j_2} over w_{j_1},
2. and w_{j_2} also prefers m_{i_1} over m_{i_2}.

That is, if both m_{i_1} and w_{j_2} would prefer each other over the partners they are currently engaged to.

The Stable Marriage Problem

Suppose that we denote the bipartition of $K_{n,n}$ by (M, W), that is, we write M for the first partite set of $K_{n,n}$ and W for the second one.

Suppose also that M represents men and W represents women, and we want to have each man from M to get 'engaged to be married' to a woman from W. **Here we have the additional assumption that each man from M has ranked all women from W in (descending) order of preference, and similarly each woman from W has ranked all men from M in (descending) order of preference.**

Clearly what we want is a perfect matching of $K_{n,n}$. Moreover...

we will consider such a matching *not stable* if it contains (at least) two pairs (m_{i_1}, w_{j_1}) and (m_{i_2}, w_{j_2}) such that

1. m_{i_1} prefers w_{j_2} over w_{j_1},
2. and w_{j_2} also prefers m_{i_1} over m_{i_2}.

That is, if both m_{i_1} and w_{j_2} would prefer each other over the partners they are currently engaged to.

If we cannot find any two such pairs in our matching, then the matching is called *stable*.

Back to a real-life interpretation of a (perfect) matching in $K_{n,n}$

Suppose one partite set of the graph represents (undergraduate or graduate) positions at Canadian universities, while the other partite set represents prospective students (and there's an edge connecting a student with a university (thus in our model with all the available positions at this university) **if and only if** this student has applied to that university).

We would like to:

- either fill all available positions,
- or have all students get accepted to one of the universities they applied to.

This corresponds to finding a matching in the graph which has maximum cardinality.

Back to a real-life interpretation of a (perfect) matching in $K_{n,n}$

Suppose one partite set of the graph represents (undergraduate or graduate) positions at Canadian universities, while the other partite set represents prospective students (and there's an edge connecting a student with a university (thus in our model with all the available positions at this university) **if and only if** this student has applied to that university).

We would like to:

- either fill all available positions,
- or have all students get accepted to one of the universities they applied to.

This corresponds to finding a matching in the graph which has maximum cardinality.

Most commonly in real-life, each university will have come up with a ranking of the students who applied to it, and also each student will have a list of preferences regarding which university to go to.

Thus, what we would really like to find here is a stable matching.

Solution to this problem: the Gale-Shapley algorithm

The algorithm will (usually) have several stages / rounds. First, we decide which partite set will be considered the 'first' set, or 'proposing' set; the other partite set will be called the 'accepting' set.

Solution to this problem: the Gale-Shapley algorithm

The algorithm will (usually) have several stages / rounds. First, we decide which partite set will be considered the 'first' set, or 'proposing' set; the other partite set will be called the 'accepting' set.

- In the first round
  - we initially have each 'man' (that is, each member of the 'proposing' partite set) propose to the 'woman' who is ranked first in his list.
  - Then, each 'woman' who has received at least one proposal replies "maybe" to the 'man' she prefers the most **out of those who proposed to her**, and "no" to all the other 'men' who proposed to her.

By the end of this round, a 'man' and a 'woman' are **provisionally** engaged if the 'man' proposed to the 'woman' and the 'woman' did reply "maybe". Also, we might still have unengaged 'men' and 'women' (and if we do, then we need to go on with the process).

Solution to this problem: the Gale-Shapley algorithm (cont.)

- In the subsequent round,
  - we initially have each **unengaged** ‘man’ propose to his most preferred ‘woman’ <u>out of those he has not yet proposed to</u>.

Solution to this problem: the Gale-Shapley algorithm (cont.)

- In the subsequent round,
  - we initially have each **unengaged** 'man' propose to his most preferred 'woman' out of those he has not yet proposed to.
  - Then, each unengaged 'woman' who has received a proposal in this round replies "maybe" to the 'man' she prefers the most out of those who proposed to her, and "no" to all the other 'men' who proposed to her,
  - but also **each already engaged 'woman'** replies "maybe" to a 'man' who proposed to her in this round **if she prefers him over the 'man' she is currently engaged to** (in such a case her previous partner is 'rejected' and becomes unengaged again).

Solution to this problem: the Gale-Shapley algorithm (cont.)

- In the subsequent round,
  - we initially have each **unengaged** 'man' propose to his most preferred 'woman' out of those he has not yet proposed to.
  - Then, each unengaged 'woman' who has received a proposal in this round replies "maybe" to the 'man' she prefers the most out of those who proposed to her, and "no" to all the other 'men' who proposed to her,
  - but also **each already engaged 'woman'** replies "maybe" to a 'man' who proposed to her in this round **if she prefers him over the 'man' she is currently engaged to** (in such a case her previous partner is 'rejected' and becomes unengaged again).

Again, by the end of the round we have some 'men' and 'women' **provisionally** engaged, and possibly some unengaged 'men' and 'women'.

Solution to this problem: the Gale-Shapley algorithm (cont.)

- In the subsequent round,
  - we initially have each **unengaged** 'man' propose to his most preferred 'woman' out of those he has not yet proposed to.
  - Then, each unengaged 'woman' who has received a proposal in this round replies "maybe" to the 'man' she prefers the most out of those who proposed to her, and "no" to all the other 'men' who proposed to her,
  - but also **each already engaged 'woman'** replies "maybe" to a 'man' who proposed to her in this round **if she prefers him over the 'man' she is currently engaged to** (in such a case her previous partner is 'rejected' and becomes unengaged again).

Again, by the end of the round we have some 'men' and 'women' **provisionally** engaged, and possibly some unengaged 'men' and 'women'.

- We repeat this process until everyone is engaged, **at which stage the engagements become final, and we are guaranteed to have found a stable matching.**

Applying this to an example

Based on the following lists of preferences, find a stable matching in $K_{5,5}$.

m_1	m_2	m_3	m_4	m_5
w_1	w_1	w_3	w_1	w_2
w_3	w_3	w_5	w_4	w_1
w_4	w_4	w_4	w_3	w_3
w_2	w_5	w_2	w_5	w_4
w_5	w_2	w_1	w_2	w_5

w_1	w_2	w_3	w_4	w_5
m_5	m_2	m_3	m_1	m_2
m_3	m_4	m_1	m_4	m_4
m_2	m_1	m_2	m_5	m_3
m_1	m_3	m_4	m_3	m_5
m_4	m_5	m_5	m_2	m_1

Applying this to an example

Based on the following lists of preferences, find a stable matching in $K_{5,5}$.

m_1	m_2	m_3	m_4	m_5
w_1	w_1	w_3	w_1	w_2
w_3	w_3	w_5	w_4	w_1
w_4	w_4	w_4	w_3	w_3
w_2	w_5	w_2	w_5	w_4
w_5	w_2	w_1	w_2	w_5

w_1	w_2	w_3	w_4	w_5
m_5	m_2	m_3	m_1	m_2
m_3	m_4	m_1	m_4	m_4
m_2	m_1	m_2	m_5	m_3
m_1	m_3	m_4	m_3	m_5
m_4	m_5	m_5	m_2	m_1

In the 1st round, and in particular the first half of it, m_1 proposes to w_1, m_2 proposes to w_1, m_3 proposes to w_3, m_4 proposes to w_1, and m_5 proposes to w_2.

Applying this to an example

Based on the following lists of preferences, find a stable matching in $K_{5,5}$.

m_1	m_2	m_3	m_4	m_5
w_1	w_1	w_3	w_1	w_2
w_3	w_3	w_5	w_4	w_1
w_4	w_4	w_4	w_3	w_3
w_2	w_5	w_2	w_5	w_4
w_5	w_2	w_1	w_2	w_5

w_1	w_2	w_3	w_4	w_5
m_5	m_2	m_3	m_1	m_2
m_3	m_4	m_1	m_4	m_4
m_2	m_1	m_2	m_5	m_3
m_1	m_3	m_4	m_3	m_5
m_4	m_5	m_5	m_2	m_1

In the 1st round, and in particular the first half of it, m_1 proposes to w_1, m_2 proposes to w_1, m_3 proposes to w_3, m_4 proposes to w_1, and m_5 proposes to w_2.

To gather all this information, we can write

$$m_1 \to w_1, \quad m_2 \to w_1, \quad m_3 \to w_3, \quad m_4 \to w_1, \quad m_5 \to w_2.$$

Applying this to an example

Based on the following lists of preferences, find a stable matching in $K_{5,5}$.

m_1	m_2	m_3	m_4	m_5
w_1	w_1	w_3	w_1	w_2
w_3	w_3	w_5	w_4	w_1
w_4	w_4	w_4	w_3	w_3
w_2	w_5	w_2	w_5	w_4
w_5	w_2	w_1	w_2	w_5

w_1	w_2	w_3	w_4	w_5
m_5	m_2	m_3	m_1	m_2
m_3	m_4	m_1	m_4	m_4
m_2	m_1	m_2	m_5	m_3
m_1	m_3	m_4	m_3	m_5
m_4	m_5	m_5	m_2	m_1

In the 1st round, and in particular the first half of it, m_1 proposes to w_1, m_2 proposes to w_1, m_3 proposes to w_3, m_4 proposes to w_1, and m_5 proposes to w_2.

To gather all this information, we can write

$$m_1 \to w_1, \quad m_2 \to w_1, \quad m_3 \to w_3, \quad m_4 \to w_1, \quad m_5 \to w_2.$$

In the second half of the 1st round, we examine whether any of the elements of the second partite set (namely the accepting set) that has received a proposal needs to make a choice. In this instance, indeed we have that w_1 has received three proposals, from m_1, m_2 and m_4. We also observe that w_1 prefers m_2 over m_1 and m_4, thus m_1 and m_4 get rejected.

Applying this to an example

Based on the following lists of preferences, find a stable matching in $K_{5,5}$.

m_1	m_2	m_3	m_4	m_5
w_1	w_1	w_3	w_1	w_2
w_3	w_3	w_5	w_4	w_1
w_4	w_4	w_4	w_3	w_3
w_2	w_5	w_2	w_5	w_4
w_5	w_2	w_1	w_2	w_5

w_1	w_2	w_3	w_4	w_5
m_5	m_2	m_3	m_1	m_2
m_3	m_4	m_1	m_4	m_4
m_2	m_1	m_2	m_5	m_3
m_1	m_3	m_4	m_3	m_5
m_4	m_5	m_5	m_2	m_1

In the 1st round, and in particular the first half of it, m_1 proposes to w_1, m_2 proposes to w_1, m_3 proposes to w_3, m_4 proposes to w_1, and m_5 proposes to w_2.

To gather all this information, we can write

$$m_1 \to w_1, \quad m_2 \to w_1, \quad m_3 \to w_3, \quad m_4 \to w_1, \quad m_5 \to w_2.$$

In the second half of the 1st round, we examine whether any of the elements of the second partite set (namely the accepting set) that has received a proposal needs to make a choice. In this instance, indeed we have that w_1 has received three proposals, from m_1, m_2 and m_4. We also observe that w_1 prefers m_2 over m_1 and m_4, thus m_1 and m_4 get rejected.

At the end of the 1st round, we have the provisional engagements

$$(m_2, w_1), \quad (m_3, w_3), \quad \text{and} \quad (m_5, w_2).$$

We also see that m_1 and m_4 from the proposing set are unengaged.

Applying this to an example (cont.)

In the 2nd round, and in particular the first half of it, we have the unengaged members of the proposing set propose to their most preferred member of the accepting set that they haven't yet proposed to.

Applying this to an example (cont.)

In the 2nd round, and in particular the first half of it, we have the unengaged members of the proposing set propose to their most preferred member of the accepting set that they haven't yet proposed to. More specifically, the unengaged m_1 is not going to propose to w_1 again, but will now propose to w_3. Similarly, m_4 proposes to w_4 this time.

Gathering the provisional engagements we already have, as well as the current proposals, we can write

$$(m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \ \text{ and } \ m_1 \to w_3, \quad m_4 \to w_4.$$

Applying this to an example (cont.)

In the 2nd round, and in particular the first half of it, we have the unengaged members of the proposing set propose to their most preferred member of the accepting set that they haven't yet proposed to. More specifically, the unengaged m_1 is not going to propose to w_1 again, but will now propose to w_3. Similarly, m_4 proposes to w_4 this time.

Gathering the provisional engagements we already have, as well as the current proposals, we can write

$$(m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \quad \text{and} \quad m_1 \to w_3, \quad m_4 \to w_4.$$

In the second half of the 2nd round, we examine whether any of the members of the accepting set has received multiple proposals, or has received a proposal while already engaged; in either case, this member of the accepting set will need to make a choice again. In this instance, indeed we have that w_3 has received a proposal from m_1 while being provisionally engaged to m_3. Since w_3 prefers m_3 over m_1, the current engagement is kept and m_1 gets rejected.

Applying this to an example (cont.)

In the 2nd round, and in particular the first half of it, we have the unengaged members of the proposing set propose to their most preferred member of the accepting set that they haven't yet proposed to. More specifically, the unengaged m_1 is not going to propose to w_1 again, but will now propose to w_3. Similarly, m_4 proposes to w_4 this time.

Gathering the provisional engagements we already have, as well as the current proposals, we can write

$$(m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \ \text{ and } \ m_1 \to w_3, \quad m_4 \to w_4.$$

In the second half of the 2nd round, we examine whether any of the members of the accepting set has received multiple proposals, or has received a proposal while already engaged; in either case, this member of the accepting set will need to make a choice again. In this instance, indeed we have that w_3 has received a proposal from m_1 while being provisionally engaged to m_3. Since w_3 prefers m_3 over m_1, the current engagement is kept and m_1 gets rejected.

At the end of the 2nd round, we have the provisional engagements

$$(m_2, w_1), \quad (m_3, w_3), \quad (m_4, w_4) \ \text{ and } \ (m_5, w_2).$$

We also see that m_1 is unengaged.

Applying this to an example (cont.)

In the 3rd round, and in particular the first half of it, we have the unengaged m_1 propose to w_4.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_2, w_1), \quad (m_3, w_3), \quad (m_4, w_4), \quad (m_5, w_2), \quad \text{and} \quad m_1 \to w_4.$$

Applying this to an example (cont.)

In the 3rd round, and in particular the first half of it, we have the unengaged m_1 propose to w_4.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_2, w_1), \quad (m_3, w_3), \quad (m_4, w_4), \quad (m_5, w_2), \ \text{ and } \ m_1 \to w_4.$$

In the second half of the 3rd round, we observe that w_4 just received a proposal while provisionally engaged to m_4. Since w_4 prefers m_1 over m_4, the current engagement is broken, w_4 becomes provisionally engaged to m_1, and m_4 becomes unengaged again.

Applying this to an example (cont.)

In the 3rd round, and in particular the first half of it, we have the unengaged m_1 propose to w_4.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_2, w_1), \quad (m_3, w_3), \quad (m_4, w_4), \quad (m_5, w_2), \text{ and } m_1 \to w_4.$$

In the second half of the 3rd round, we observe that w_4 just received a proposal while provisionally engaged to m_4. Since w_4 prefers m_1 over m_4, the current engagement is broken, w_4 becomes provisionally engaged to m_1, and m_4 becomes unengaged again.

At the end of the 3rd round, we have the provisional engagements

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \text{ and } (m_5, w_2).$$

We also see that m_4 is unengaged.

Applying this to an example (cont.)

In the 4th round, and in particular the first half of it, we have the unengaged m_4 propose to w_3.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \quad \text{and} \quad m_4 \to w_3.$$

Applying this to an example (cont.)

In the 4th round, and in particular the first half of it, we have the unengaged m_4 propose to w_3.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \quad \text{and} \quad m_4 \to w_3.$$

In the second half of the 4th round, we observe that w_3 is already provisionally engaged to m_3, and prefers m_3 over m_4. Thus, the current engagement is kept, while m_4 remains unengaged.

Applying this to an example (cont.)

In the 4th round, and in particular the first half of it, we have the unengaged m_4 propose to w_3.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \quad \text{and} \quad m_4 \to w_3.$$

In the second half of the 4th round, we observe that w_3 is already provisionally engaged to m_3, and prefers m_3 over m_4. Thus, the current engagement is kept, while m_4 remains unengaged.

At the end of the 4th round, we have the provisional engagements

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad \text{and} \quad (m_5, w_2),$$

and m_4 is unengaged.

In the 5th round, and in particular the first half of it, we have the unengaged m_4 propose to w_5.

Applying this to an example (cont.)

In the 4th round, and in particular the first half of it, we have the unengaged m_4 propose to w_3.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \quad \text{and} \quad m_4 \to w_3.$$

In the second half of the 4th round, we observe that w_3 is already provisionally engaged to m_3, and prefers m_3 over m_4. Thus, the current engagement is kept, while m_4 remains unengaged.

At the end of the 4th round, we have the provisional engagements

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad \text{and} \quad (m_5, w_2),$$

and m_4 is unengaged.

In the 5th round, and in particular the first half of it, we have the unengaged m_4 propose to w_5.

In the second half of the 5th round, we observe that w_5 has not received any other proposals, so the proposal of m_4 is accepted.

Applying this to an example (cont.)

In the 4th round, and in particular the first half of it, we have the unengaged m_4 propose to w_3.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \quad \text{and} \quad m_4 \to w_3.$$

In the second half of the 4th round, we observe that w_3 is already provisionally engaged to m_3, and prefers m_3 over m_4. Thus, the current engagement is kept, while m_4 remains unengaged.

At the end of the 4th round, we have the provisional engagements

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad \text{and} \quad (m_5, w_2),$$

and m_4 is unengaged.

In the 5th round, and in particular the first half of it, we have the unengaged m_4 propose to w_5.

In the second half of the 5th round, we observe that w_5 has not received any other proposals, so the proposal of m_4 is accepted.

At the end of the 5th round, we have the provisional engagements

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_4, w_5) \quad \text{and} \quad (m_5, w_2).$$

Applying this to an example (cont.)

In the 4th round, and in particular the first half of it, we have the unengaged m_4 propose to w_3.

Gathering the provisional engagements we already have, as well as the current proposal, we can write

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_5, w_2), \quad \text{and} \quad m_4 \to w_3.$$

In the second half of the 4th round, we observe that w_3 is already provisionally engaged to m_3, and prefers m_3 over m_4. Thus, the current engagement is kept, while m_4 remains unengaged.

At the end of the 4th round, we have the provisional engagements

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad \text{and} \quad (m_5, w_2),$$

and m_4 is unengaged.

In the 5th round, and in particular the first half of it, we have the unengaged m_4 propose to w_5.

In the second half of the 5th round, we observe that w_5 has not received any other proposals, so the proposal of m_4 is accepted.

At the end of the 5th round, we have the provisional engagements

$$(m_1, w_4), \quad (m_2, w_1), \quad (m_3, w_3), \quad (m_4, w_5) \quad \text{and} \quad (m_5, w_2).$$

Given that every member of the proposing set (and hence every member of the accepting set too) is now engaged, these engagements become final. **We can also check that the matching they form is stable.**

A (more 'troublesome'?) variation of the previous example

Suppose we have the following lists of preferences. Find a stable matching (again, view the first set as the proposing set).

m_1	m_2	m_3	m_4	m_5
w_1	w_1	w_1	w_1	w_1
w_2	w_2	w_2	w_2	w_2
w_5	w_5	w_5	w_5	w_5
w_3	w_3	w_3	w_3	w_3
w_4	w_4	w_4	w_4	w_4

w_1	w_2	w_3	w_4	w_5
m_5	m_5	m_3	m_3	m_2
m_4	m_4	m_1	m_2	m_1
m_3	m_3	m_2	m_1	m_5
m_2	m_2	m_4	m_5	m_4
m_1	m_1	m_5	m_4	m_3

An actual real-life application of the algorithm (and of even more detailed variants of it)

The Gale-Shapley algorithm and refinements of it are put to practice every year in the USA by the **National Resident Matching Program**, sometimes more simply called the **Match**.

This is an organisation that runs the process during which students who have finished medical school apply to different recidency training programmes offered by US teaching hospitals, and are subsequently placed into one of those programmes.
— Before a matching is reached, each training hospital invites some of the students who have applied to their programme for an interview, and then submits a list ranking those candidates.
— Similarly, each student submits a list with their preferences.

Note that there is no requirement that these lists are full (in other words, a teaching hospital may only rank the candidates that they interviewed, and similarly a student may only rank a handful of programmes and not express any preference regarding the rest). Of course this increases the chances that the announced matching will not be perfect, and hence either some programmes will not have all their available positions filled, or some students will not be placed into any programme.

That said, the announced matching will still be stable.

For more details, see e.g. the site: `https://www.nrmp.org/` of NRMP,
as well as a brief YouTube introduction by NRMP at
`https://staging-nrmp.kinsta.cloud/matching-algorithm/` .

Another real-life problem that can be answered via Graph Theory

Scheduling exams:

Another real-life problem that can be answered via Graph Theory

Scheduling exams: Assume that the Registrar's Office of our university wants to come up with the final exam schedule for the Fall term.

The main restriction that they have to pay attention to is that classes which have common students enrolled should not have their exams scheduled at the same time.

Otherwise, if the student rosters of two courses have an empty intersection, then the final exams of these two courses can take place concurrently (and of course it would be desirable to have multiple exams running at the same time in order to have only a few days of final exams).

Another real-life problem that can be answered via Graph Theory

Scheduling exams: Assume that the Registrar's Office of our university wants to come up with the final exam schedule for the Fall term.

The main restriction that they have to pay attention to is that classes which have common students enrolled should not have their exams scheduled at the same time.

Otherwise, if the student rosters of two courses have an empty intersection, then the final exams of these two courses can take place concurrently (and of course it would be desirable to have multiple exams running at the same time in order to have only a few days of final exams).

If we consider a graph with vertex set all the different courses offered in the Fall term, and we assume that there is an edge joining two courses **if and only if their student rosters have a non-empty intersection**, what kind of vertex subsets would we be interested in, that would correspond to groups of courses whose exams can run at the same time?

Another real-life problem that can be answered via Graph Theory

Scheduling exams: Assume that the Registrar's Office of our university wants to come up with the final exam schedule for the Fall term.

The main restriction that they have to pay attention to is that classes which have common students enrolled should not have their exams scheduled at the same time.

Otherwise, if the student rosters of two courses have an empty intersection, then the final exams of these two courses can take place concurrently (and of course it would be desirable to have multiple exams running at the same time in order to have only a few days of final exams).

If we consider a graph with vertex set all the different courses offered in the Fall term, and we assume that there is an edge joining two courses **if and only if their student rosters have a non-empty intersection**, what kind of vertex subsets would we be interested in, that would correspond to groups of courses whose exams can run at the same time?

Answer. We are trying to partition / 'break up' the vertex set (that is, the entire course roster of the Fall term) into independent sets of vertices (that is, subsets of courses which do not have students in common).

Another real-life problem that can be answered via Graph Theory

Scheduling exams: Assume that the Registrar's Office of our university wants to come up with the final exam schedule for the Fall term.

The main restriction that they have to pay attention to is that classes which have common students enrolled should not have their exams scheduled at the same time.

Otherwise, if the student rosters of two courses have an empty intersection, then the final exams of these two courses can take place concurrently (and of course it would be desirable to have multiple exams running at the same time in order to have only a few days of final exams).

If we consider a graph with vertex set all the different courses offered in the Fall term, and we assume that there is an edge joining two courses **if and only if their student rosters have a non-empty intersection**, what kind of vertex subsets would we be interested in, that would correspond to groups of courses whose exams can run at the same time?

Answer. We are trying to partition / 'break up' the vertex set (that is, the entire course roster of the Fall term) into *(as few as possible)* independent sets of vertices (that is, subsets of courses which do not have students in common).

Such partitions will be called 'vertex colourings'

For the following definition, we will be using the set

$$\mathbb{N}_+ = \{1, 2, 3, \ldots\}$$

of positive integers. We will be thinking of each positive integer as a different **colour** (even though most of the time we will keep unspecified which integer corresponds to which colour).

Such partitions will be called 'vertex colourings'

For the following definition, we will be using the set

$$\mathbb{N}_+ = \{1, 2, 3, \ldots\}$$

of positive integers. We will be thinking of each positive integer as a different **colour** (even though most of the time we will keep unspecified which integer corresponds to which colour).

Definition

Let $G = (V, E)$ be a graph. A *vertex colouring* of G is any function

$$\xi : V(G) \to \mathbb{N}_+.$$

Such partitions will be called 'vertex colourings'

For the following definition, we will be using the set

$$\mathbb{N}_+ = \{1, 2, 3, \ldots\}$$

of positive integers. We will be thinking of each positive integer as a different **colour** (even though most of the time we will keep unspecified which integer corresponds to which colour).

Definition

Let $G = (V, E)$ be a graph. A *vertex colouring* of G is any function

$$\xi : V(G) \to \mathbb{N}_+.$$

The vertex subsets $V_i = \{v \in V(G) : \xi(v) = i\}$ will be called the *colour classes* of the vertex colouring ξ. Observe that the **non-empty** colour classes form a partition of $V(G)$.

Such partitions will be called 'vertex colourings'

For the following definition, we will be using the set

$$\mathbb{N}_+ = \{1, 2, 3, \dots\}$$

of positive integers. We will be thinking of each positive integer as a different **colour** (even though most of the time we will keep unspecified which integer corresponds to which colour).

Definition

Let $G = (V, E)$ be a graph. A *vertex colouring* of G is any function

$$\xi : V(G) \to \mathbb{N}_+.$$

The vertex subsets $V_i = \{v \in V(G) : \xi(v) = i\}$ will be called the *colour classes* of the vertex colouring ξ. Observe that the **non-empty** colour classes form a partition of $V(G)$.

— A vertex colouring of G is called a *proper (vertex) colouring* if

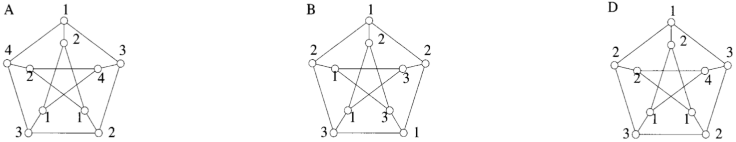
no two adjacent vertices belong to the same colour class.

In other words, if each of the colour classes is an independent set of vertices.

‘Drawing’ vertex colourings

Despite what the term suggests, not every vertex colouring has to be represented by actually colouring the vertices, even though this is always an option if it’s practical (that is, if not many colours appear in the range of the colouring map).

Alternatively, we can write next to each vertex the integer it is mapped to *(see following image containing different vertex colourings of the Petersen graph).*

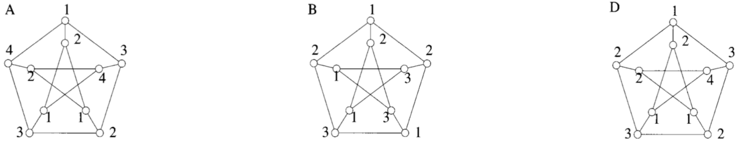


from Wallis’ book

'Drawing' vertex colourings

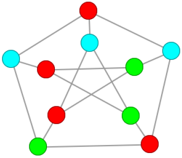
Despite what the term suggests, not every vertex colouring has to be represented by actually colouring the vertices, even though this is always an option if it's practical (that is, if not many colours appear in the range of the colouring map).

Alternatively, we can write next to each vertex the integer it is mapped to *(see following image containing different vertex colourings of the Petersen graph).*



from Wallis' book

Of course, vertex colouring B (for instance) can also be clearly conveyed in this way:



Chromatic number of a graph

From now on we focus almost exclusively on proper colourings of graphs.

Definition 1

Let $G = (V, E)$ be a graph. A proper vertex colouring ξ of G is called an *n-colouring* if there are exactly n non-empty colour classes of ξ. In other words, if the range of the function ξ contains exactly n positive integers.

Chromatic number of a graph

From now on we focus almost exclusively on proper colourings of graphs.

Definition 1

Let $G = (V, E)$ be a graph. A proper vertex colouring ξ of G is called an *n-colouring* if there are exactly n non-empty colour classes of ξ. In other words, if the range of the function ξ contains exactly n positive integers.

G will be called n-colourable if we can find a (proper) n-colouring of G *(pictorially we can think of this as follows: G is n-colourable if n colours are enough for us to find a way to colour the vertices of G so that no two adjacent vertices will end up having the same colour)*.

Chromatic number of a graph

From now on we focus almost exclusively on proper colourings of graphs.

Definition 1

Let $G = (V, E)$ be a graph. A proper vertex colouring ξ of G is called an *n-colouring* if there are exactly n non-empty colour classes of ξ. In other words, if the range of the function ξ contains exactly n positive integers.

G will be called n-colourable if we can find a (proper) n-colouring of G *(pictorially we can think of this as follows: G is n-colourable if n colours are enough for us to find a way to colour the vertices of G so that no two adjacent vertices will end up having the same colour)*.

Definition 2

The *chromatic number* of a graph G is equal to the **smallest** integer n for which we can find a (proper) n-colouring of G.

If n_0 is this smallest integer, then we say that G is *n_0-chromatic*.

Chromatic number of a graph

From now on we focus almost exclusively on proper colourings of graphs.

Definition 1

Let $G = (V, E)$ be a graph. A proper vertex colouring ξ of G is called an *n-colouring* if there are exactly n non-empty colour classes of ξ. In other words, if the range of the function ξ contains exactly n positive integers.

G will be called n-colourable if we can find a (proper) n-colouring of G *(pictorially we can think of this as follows: G is n-colourable if n colours are enough for us to find a way to colour the vertices of G so that no two adjacent vertices will end up having the same colour)*.

Definition 2

The *chromatic number* of a graph G is equal to the **smallest** integer n for which we can find a (proper) n-colouring of G.

If n_0 is this smallest integer, then we say that G is *n_0-chromatic*. We denote this smallest integer by $\chi(G)$ *(in other words, G is $\chi(G)$-chromatic)*.

Chromatic number of a graph

From now on we focus almost exclusively on proper colourings of graphs.

Definition 1

Let $G = (V, E)$ be a graph. A proper vertex colouring ξ of G is called an *n-colouring* if there are exactly n non-empty colour classes of ξ. In other words, if the range of the function ξ contains exactly n positive integers.

G will be called n-colourable if we can find a (proper) n-colouring of G *(pictorially we can think of this as follows: G is n-colourable if n colours are enough for us to find a way to colour the vertices of G so that no two adjacent vertices will end up having the same colour)*.

Definition 2

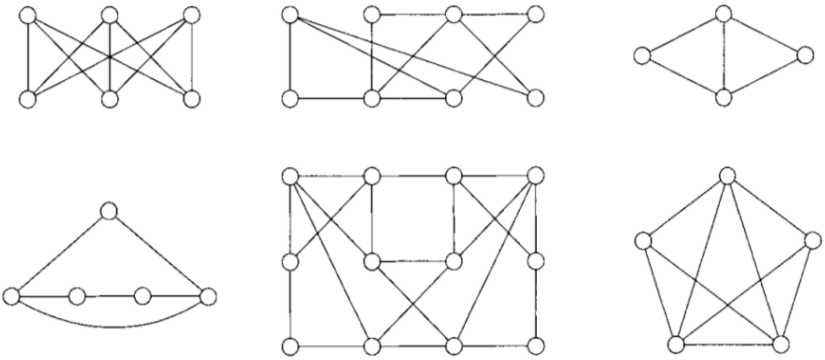
The *chromatic number* of a graph G is equal to the **smallest** integer n for which we can find a (proper) n-colouring of G.

If n_0 is this smallest integer, then we say that G is *n_0-chromatic*. We denote this smallest integer by $\chi(G)$ *(in other words, G is $\chi(G)$-chromatic)*.

Finally, a $\chi(G)$-colouring of G, that is, a proper colouring of G in $\chi(G)$ colours, is called *minimal*.

Vertex colouring in some examples

Practice Exercise. For each of the following graphs, find its chromatic number, and also give a minimal colouring.



from Wallis' book