Matchings and Edge Colourings

Complement to Chapter 5, “The Unhappy Employee”
and Chapter 9, “The Sudoku Apprentice”

Let’s consider *n* hockey teams that will be playing against each other in a tournament. Let’s suppose that each team needs to play each other team exactly once. We can start by asking: what is the maximum number of games that can take place each day, knowing that no team can play two games on the same day?

The answer is simple: if *n* is even, then *n*/2 games can take place on the same day; if *n* is odd, then the number is reduced to (*n*-1)/2, which means that at least one team is not playing each day.

In terms of a graph, we can create one vertex per team and link each pair of vertices with an edge. The edges correspond, then, to the games that need to be planned. Our question then comes down to asking what is the maximum number of edges we can choose in the graph without any two edges chosen having any ends in common. What we’re looking for is in fact called a *matching* with a maximum number of edges.

Definition

A “matching” in a graph is a set of edges that have no ends in common.

We can also ask what the minimum number of days necessary is so that all the games can take place. Given that each team plays each other team exactly once, we need to plan *n*(*n*-1)/2 games.

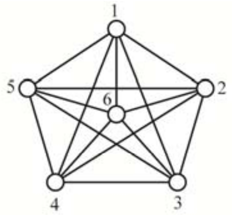
- If *n* is even, we have seen that a maximum of *n*/2 games can take place each day and at least *n*-1 days are therefore necessary so that all the games can be played.
- If *n* is odd, we have seen that a maximum of (*n*-1)/2 games can take place each day, which means that at least *n* days are needed for all the games to be held.

As for a graph, this means we want to colour the edges of the graph using as few colours as possible, such that each colour represents a matching. In other words, we’re associating one colour with each day.

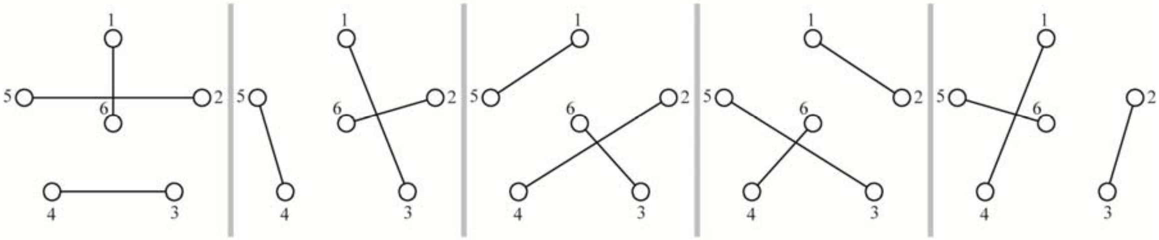
It is easy to plan all the *n*(*n*-1)/2 games in *n*-1 days when *n* is even. We can do this as follows. We number the vertices from 1 to *n* to represent the *n* teams. Let’s draw the graph described above, putting the *n* vertex in the centre and the other vertices in a circle around vertex *n*.

- On the first day, we organize a game between teams 1 and *n*, 2 and *n*-1, 3 and *n*-2, and so forth until the game between *n*/2 and *n*/2+1.
- On the following days, we reproduce what happened the day before, but we simply rotate the matchings clockwise.

Since a picture is worth a thousand words, here is an illustration of this construction for a set of *n*=6 teams.



The 15 edges correspond
to 15 games to be planned



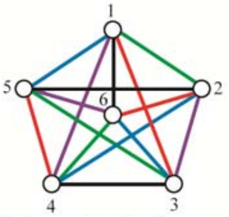
First day
(black)

Second day
(green)

Third day
(blue)

Fourth day
(green)

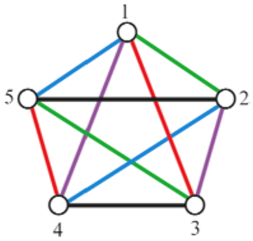
Fifth day
(purple)



Full schedule for five days

In the case where *n* is odd, it’s now also easy to plan. We just need to add a non-existent team, which we’ll note as *n*+1. When a team plays against team *n*+1, that just means the team is at rest. So we now have a schedule to build with *n*+1 even, which requires (*n*+1)-1 days, which is *n* days. We have seen that we can’t do better. To obtain the schedule of games for the *n* teams, we simply need to remove everything related to the non-existent team *n*+1.

For example, if the tournament brings together *n*=5 teams, we add a sixth non-existent team and we build the schedule described above. Then we erase everything concerning the sixth team to get the following schedule for five days and five teams.



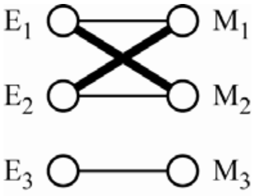
Edge matchings and colourings are useful in many other situations too. Let's suppose, for instance, that we need to carry out five tasks, T_1, T_2, T_3, T_4 and T_5, with each of the tasks taking one day to complete. Tasks T_1 and T_2 must be performed by employee E_1, tasks T_3 and T_4 by employee E_2 and task T_5 by employee E_3. Tasks T_1 and T_3 require machine M_1, tasks T_2 and T_4 need machine M_2 and task T_5 machine M_3. Knowing that each employee can only carry out one task at a time and that each machine can only be used by one employee at a time,

- How many tasks can be performed at maximum in one day?
- How many days are needed at minimum to perform the five tasks?

We can of course generalize this problem to any number of tasks, employees and machines.

The two problems can be modelled using a graph. The vertices are employees E_1, E_2 and E_3 as well as machines M_1, M_2 and M_3. We link a vertex E_i to a machine M_j if a task requires employee E_i and machine M_j. The five tasks in our example are therefore represented with five edges. To answer the first question, we need to determine a matching with a maximum number of edges in the graph. In our example, the biggest matching includes three edges. We can choose $[E_1,M_1]$, $[E_2,M_2]$ and $[E_3,M_3]$, which corresponds with tasks T_1, T_4 and T_5. We could also have chosen $[E_1,M_2]$, $[E_2,M_1]$ and $[E_3,M_3]$, which would have corresponded with tasks T_2, T_3 and T_5.

The second problem is equivalent to colouring the edges of this same graph using a minimum number of colours, such that the edges that touch one another are of different colours. Each colour must correspond to a matching (not necessarily maximum) and represent what will be done on a given day. In our example, we need two days to carry out the five tasks. For example, we can perform tasks T_1, T_4 and T_5 on the first day (fine edges) and tasks T_2 and T_3 the second day (bold edges).

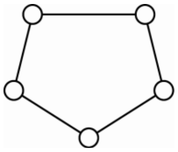


Definition

Colouring the edges of a graph G means assigning colours to the edges such that the edges with ends in common are of different colours. We are also generally seeking to determine a colouring using as few colours as possible. The smallest number of colours needed to colour the edges of a graph G is called the "chromatic index" of G and is noted as q(G).

We note $\Delta(G)$ for the highest degree of a vertex in G. Given that all the edges touching a given vertex must be of different colours, it is clear that the chromatic index q(G) of G cannot be less than $\Delta(G)$. It is possible, however, that q(G) may be strictly higher than

$\Delta(G)$. For example, the edges of the pentagon below cannot be coloured using fewer than three colours, while $\Delta(G)=2$.



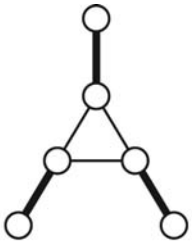
In 1964, Vizing demonstrated that the chromatic index q(G) is never much higher than the highest degree $\Delta(G)$.

Theorem (Vizing)

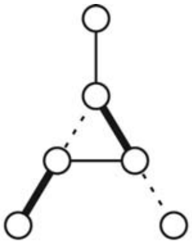
The following inequalities are valid for all graphs G: $\Delta(G) \leq q(G) \leq \Delta(G)+1$.

Vizing even provided a procedure for determining the colouring of the edges for graph G in $\Delta(G)+1$ colours. His algorithm is in error, eventually, by one unit on the chromatic index, which is not enormous. It is, however, very difficult to know whether $q(G)=\Delta(G)$ or $\Delta(G)+1$ when G has no particular properties.

To try and determine q(G), we can for instance determine a first matching with a maximum of edges in G, assign a colour to this matching, and remove it from the graph. By repeating this process, we will colour all the edges of the graph, and we hope to use as few colours as possible, since at every step we are seeking the highest number of edges that can all be of the same colour. This way of operating, however, does not always colour G in q(G) colours. For example, in the graph below, the biggest matching contains three edges: they are the three edges touching the 1-degree vertices. By giving the same colour to these three edges, we are still left with the triangle to colour, which requires three new colours, since the three edges all have ends in common. The procedure described above thus gives a total of four colours, while the chromatic index q(G) is 3, as illustrated below with the plain, bold and dotted lines.



Graph whose biggest matching includes three edges (bold)



The chromatic index q(G) is 3

In some cases, it is nevertheless easy to know whether $q(G)=\Delta(G)$ or $\Delta(G)+1$. This is the case, for instance, in complete graphs where no edges are missing.

Definition

A graph G is "complete" if there is an edge linking every pair of vertices.

Earlier, we saw that the planning of a tournament for *n* teams is equivalent to colouring the edges of a complete graph with *n* vertices. We saw that *n*-1 days would suffice if *n* is even, while we need *n* days if *n* is odd. Formally speaking, we saw the following property.

Property

In a complete graph G with *n* vertices,

- q(G) = Δ(G) = *n*-1 if *n* is even
- q(G) = Δ(G)+1 = *n* if *n* is odd

Another easy case is one in which the vertices of the graph are partitioned into two sets, V_1 and V_2, such that every edge in the graph has exactly one end in V_1 and the other in $V_{2.}$

Definition

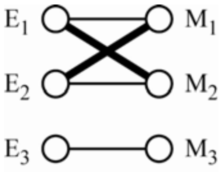
A graph G is "bipartite" if there exists a partition (V_1, V_2) of the vertices such that each edge in G has one end in V_1 and the other in V_2.

The task planning example we saw earlier consists of colouring the edges of a bipartite graph. The partition of the vertices is simple: we can place the employees in set V_1 and the machines in set V_2. The edges representing the tasks always link one employee to one machine.

Property

If G is a bipartite graph, then q(G) = Δ(G).

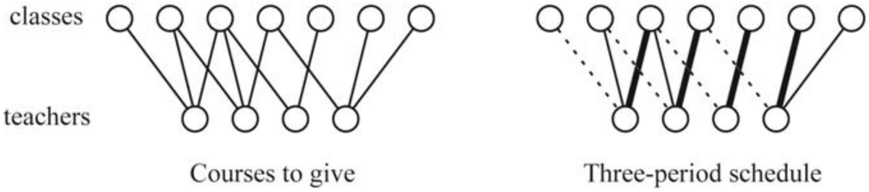
In the graph for the five tasks, which is reproduced here, the highest degree is reached by vertices E_1, E_2, M_1 and M_2, and it is 2. The chromatic index is therefore 2, which we represent using regular and bold lines.



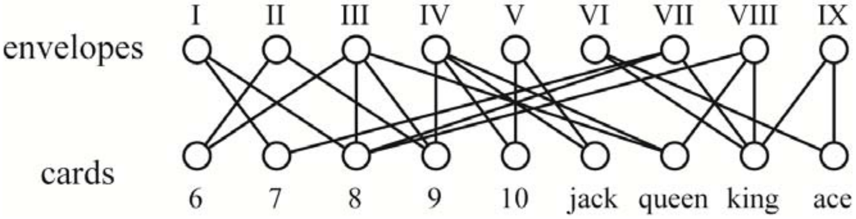
Another example

In a Montréal college, *n* teachers need to give courses to *m* classes (groups of students). Each edge of the left-hand graph below corresponds to one course to be given. Knowing that a teacher cannot give two courses at the same time, and that a class cannot take two courses at the same time, how many periods do we need to set out in the schedule so that all the courses can be taught?

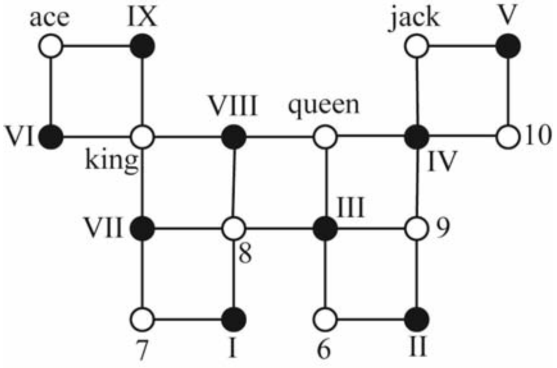
This is a colouring problem for the edges of a bipartite graph. The number of periods required is then q(G)=Δ(G). In our case, this number is 3, and a three-colour schedule is set out on the right with bold, regular and dotted lines.



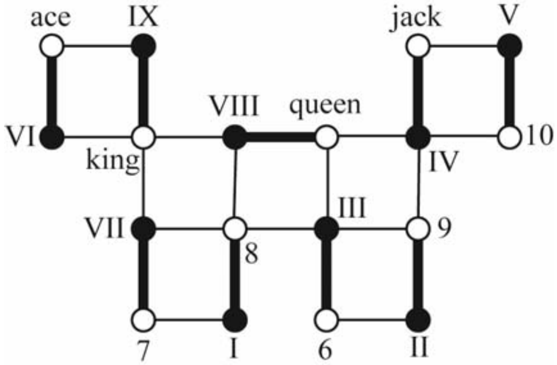
Let's come back for a moment to our quick-on-the-draw man, Inspector Manori. To give Cindy back her smile, he does a magic trick for her in which he asks her to place nine cards into nine envelopes. Each envelope can only contain one of the cards authorized by Manori. The situation is represented using the following bipartite graph, in which we need to determine a matching that touches all the vertices.

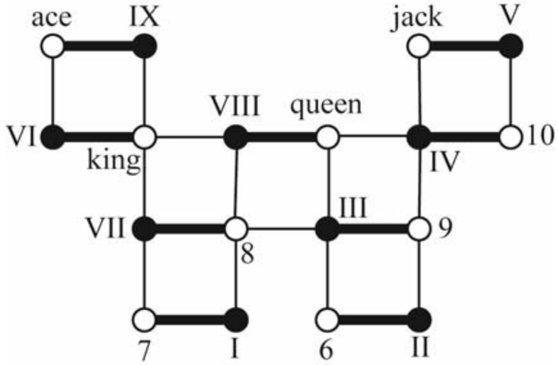


By moving the vertices around to avoid edge crossovers, Manori succeeds in providing the following representation of the same graph, the bipartition represented this time using black and white colourings for the vertices—black for the envelopes and white for the cards.

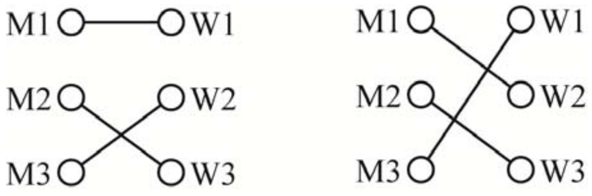


Manori shows Cindy that all 9-edge matchings in this graph necessarily contain the edge linking vertex VIII to the queen vertex. He shows her two of the matchings.



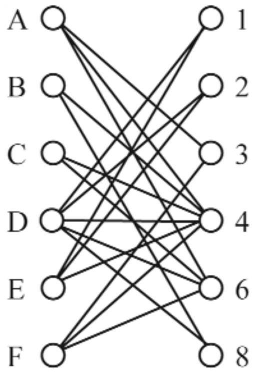


Before meeting Cindy, Manori has a big discussion with Despontin during which they talk about stable marriages. A set of marriages *n* between *n* men and *n* women is once again a matching in a bipartite graph with the men in V_1 and the women in V_2. Here are two examples of the matchings he provides for three men and three women.



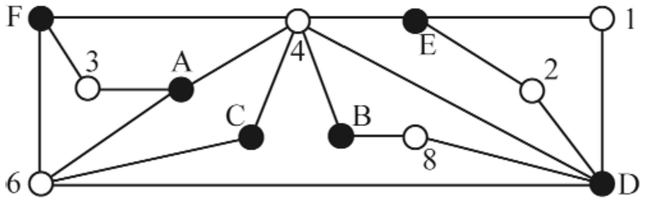
We conclude this chapter with the second Sudoku grid that the young Lei asks Manori to fill out. To do so, Manori chooses six specific boxes in the grid and observes that these boxes can only contain the numbers 1, 2, 3, 4, 6 and 8. So he builds a bipartite graph with the boxes as the first set of vertices, V_1, and the available numbers as the second set of vertices, V_2. He links a vertex from V_1 to a vertex from V_2 if the number of V_2 can be placed in the box of V_1. He gets the following graph.

	2	A	9				8	
9		B				6		3
1	7	C	8		3			
		5						
7	3	D						
	9	E		8			6	
		F		2	7	1	9	8



To fill out the six boxes, we need to determine a matching in the bipartite graph, since each box can only contain one of the six numbers, and each number can only go in one of the six boxes (as they are all in the same column).

To get a better view of things, Manori decides to move the vertices around a bit and redraw them such that there are no more edge crossovers. He obtains the following new representation, in which the boxes of V_1 are black and the numbers of V_2, to be placed in the six boxes, are white.



We can clearly see that this graph is bipartite because there is no edge linking any two white vertices or black vertices. The graph is also a planar plane graph because none of its edges cross.

Manori observes that all the six-edge matchings in this bipartite graph necessarily contain the edge linking B to 8, which means that we can place the number 8 in the B box.