Matchings and flows

ICTS-RRI Maths Circle, Bengaluru

12, 26 April 2025, 10:00 am to 1:00 pm

10, 24 May 2025, 10:00 am to 1:00 pm

Session I: 12 April 2025

Matchings

Employees and tasks: On a certain morning, n employees show up at an office. The manager determines that there are m tasks that need to be completed that day. To complete a task an employee must be assigned to work on it for the entire day. Tasks need differing skills—not all employees possess the necessary training to perform all tasks. The manager knows the tasks that each employee can perform (see fig. 1). How can the manager determine if all m tasks can still be completed that day?

	t_1	t_2	t_3	t_4	t_5
w_1			+		
w_2	+	+			
w_3		+	+		
w_4		+	+		
w_5			+	+	+

Figure 1: Employees (w_1 to w_5) and tasks (t_1 to t_5)

Cabs and customers: At a certain time, a cab company has n taxis currently available; the company knows where each cab is located. The company also has requests from m customers, who would each like a cab. The company can determine (based on current traffic conditions) how long a particular cab will take to reach a particular customer. Can the cabs be assigned so that no customer needs to wait more than five minutes?

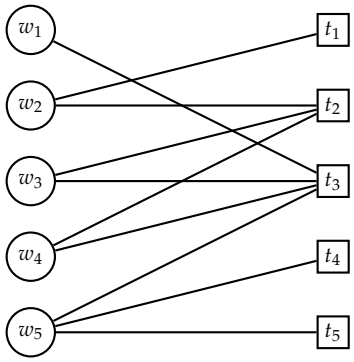

Figure 2: The graph for the office

Terminology

Problems such as these are called matching problems, or more precisely, *bipartite matching* problems, for there are two kinds of entities in each problem: employees-tasks and cabs-customers. Such problems are often visualised using *bipartite graphs*.

Bipartite graphs have two sets of vertices (nodes): A and B. In the first example, we take A to be the set of employees and B to be the set of tasks; in the second, we take A to be the set of cabs and B to be the set of customer requests. If vertex a in A is *compatible* with a vertex b in B then we place an edge (a, b) between them; e.g., we place an edge between employee w_4 and task t_3 because w_4 has been trained to for task t_3. The *input* to a matching problem is such a graph consisting of the vertex sets A and B, and the set of edges E: it is customary to write the three together as (A, B, E) and refer to this triple as the bipartite graph, and give it a name, such as G, G', H, etc.[1] There are other ways to represent graphs, of course. For example, we could draw a picture with the vertices in A on the left, and

$$
\begin{aligned}
&A = \{w_1, w_2, w_3, w_4, w_5\};\\
&B = \{t_1, t_2, t_3, t_4, t_5\};\\
&E = \{(w_1, t_3), (w_2, t_1), (w_2, t_2), (w_3, t_2), (w_3, t_3),\\
&\quad (w_4, t_2), (w_4, t_3), (w_4, t_3), (w_4, t_4), (w_4, t_5)\}.
\end{aligned}
$$

$$
M_G = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.
$$

Figure 3: The vertex set, the edge set, and the adjacency matrix

[1] The meaning of the term *graph* here is different from the more familiar notion associated with functions. In an abstract sense, the two are related, but that will not be important our discussion.

the vertices in B on the right, and connect a and b by a line whenever (a, b) is an edge of the graph. The lines need not be straight, the intermediate points hold no meaning, only the end points do, so we don't need to ensure that they do not cross; of course, as with every other diagram, keeping the drawings neat and symmetrical is often helpful. We will sometimes allow multiple edges (in parallel) between the same pair of vertices. Or one could represent the graph using a matrix, where the rows are for vertices in A and the columns are for vertices in B, and the entry in row a and column b is 1 if (a, b) is an edge, and 0 otherwise. This is the *adjacency matrix* of the graph—we write M_G for the adjacency matrix of G; note that M_G depends on the order we choose for the vertices of A and B. The entries of the adjacency matrices can be manipulated as numbers, by multiplying them, adding them, etc.; as a result what started off as combinatorial information takes on an algebraic life with many deep and beautiful consequences, which we will discuss some other time. Of these many ways of describing a graph, we will find the picture representation the most useful for our discussion. The *degree of a vertex* a is the number of edges that have a as an end-point; we write $\deg(a)$ to refer to the degree of a. In the graph of fig. 2, $\deg(w_2) = 2$.

A *matching* in a bipartite graph G is a set of disjoint edges. If (a, b) is an edge in the matching, then we say that a is matched to b. For example, in the graph of fig. 2 the set of edges

$$X = \{(w_4, t_2), (w_3, t_3), (w_5, t_4)\}.$$

is a matching, because no two edges in this set share a vertex. A *maximum cardinality matching* or just *maximum matching* is a matching with the most edges. A *perfect matching* is a matching where every vertex of the graph are matched.

Exploration I: The card puzzle

Take the standard deck of cards with four suits (spades, hearts, diamonds, clubs) and with 13 cards of each suit with ranks A, 2, . . . , 9, 10, J, K, Q. Shuffle the deck if you wish. Now arrange the cards in 13 rows of 4 cards each. For example, consider the arrangement of cards shown in fig. 4.

I.1 Can you pick one card from each row so that you get all 13 ranks (they need not be of the same suit, of course)? (This might take some trial and error.) Try to model this as a matching problem.

I.2 What is the bipartite graph (A, B, E) corresponding to this problem? What do A and B represent? When would you connect a vertex a in A and a vertex b in B by an edge? Does your graph

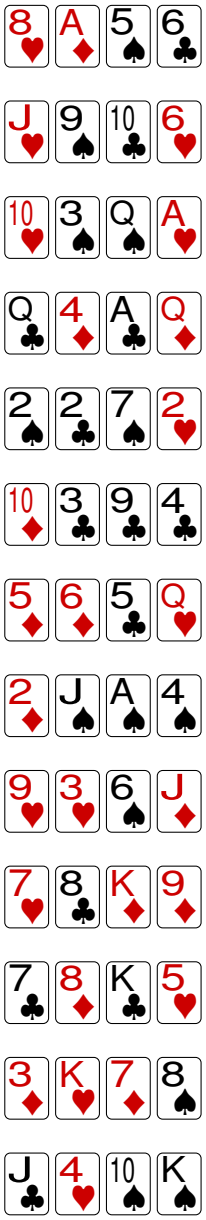

Figure 4: The card puzzle

have parallel edges? What are the degrees of the vertices in A and B?

I.3 Show that in every bipartite graph where all vertices have the same degree, say $r \geq 1$, both A and B must have the same number of vertices? Such graphs are said to be r-regular.

I.4 Make up some r-regular bipartite graphs ($r \geq 1$). Does each of them have a perfect matching?

Exploration II: Greedy

Is there a systematic way to find a maximum matching in a given bipartite graph? Let us try the greedy matching strategy shown in fig. 5.

II.1 Find a bipartite graph where this greedy method does not find the maximum matching.

II.2 How bad is the solution of the greedy method? Show that the greedy method always finds a matching that is at least half as large as the best, that is,

$$|\text{Greedy}(G)| \geq \frac{1}{2}|\text{Opt}(G)|.$$

Step 0: List the edges in some order: $e_1, e_2, \ldots, e_m$; start from the empty matching $X = \{\}$.

Step 1: Add e_1 to X.

Step 2: If e_2 is disjoint from e_1, add e_2 to X.

⋮ ⋮

Step ℓ: If e_ℓ is disjoint from the edges already in X, add e_ℓ to X.

Step m: If e_m is disjoint from the edges already in X, add e_m to X.

Final step: Output X.

Figure 5: Greedy

Exploration III: Beyond greedy

Let us try another strategy. Recall that the greedy method maintains a matching X and tries to improve it (i.e., make it bigger) by adding another edge if it is not in conflict with an edge already in X. What else could one do? Look carefully at Greedy's behaviour in the example you constructed in Exploration II (i)? What did it miss?

To help our thoughts, let us use the following terminology. Suppose we have a bipartite graph G and a matching X, which we want to improve.

Matched vertex, free vertex: A vertex v is a *matched vertex* (or saturated vertex) with respect to X if it is and end points of an edge in X; v is a *free vertex* (unmatched vertex) if it is not a matched vertex.

Matching edge: An edge (a, b) is a matching edge if (a, b) is in X.

X saturates S: We say that X saturates a set S of vertices if every vertex in S is matched with respect to X.

Is *adding* an edge the only way to improve X? We are not able to add more edges because we are being obstructed by the edges already in X? Could we, perhaps, remove one edge from X, and create an opportunity for adding two? That too would result in a bigger matching. What else? After working out some examples, write down a new strategy; state it in a generally applicable form.

Figure 6: Greedier than greedy?

The greedy method when it makes progress ends up converting two free vertices into matched vertices. By looking at examples, see if you can discover a less short-sighted version of the greedy method, that succeeds where the greedy method fails.

Report on the session of 12 April

Everybody seemed to be comfortable with the various representations of the matching problem. I mentioned that it is often convenient to allow parallel edges; this almost immediately provoked the question about how such graphs would be represented as matrices (answer: if there are k edges connecting i and j, then we write k in position (i, j) of the matrix). After introducing the terminology, we move on to the first exploration, the card puzzle. With some trial and error, everybody found a solution. We drew the graph and noticed that all vertices had four edges, so the graph was 4-regular.*Why do such regular graphs always have a matching?*–this was the first question we wanted to study. Second, *Is there a systematic method for finding the matching?* With these questions at the back of our mind, we moved on to Exploration II. That the two parts in an r-regular ($r \geq 1$) bipartite graph must have the same number of vertices was immediate for most. Somehow, against my expectations, the next part was very confusing for the majority. I drew a picture, where the greedy matching connected $S_1 \subseteq A$ and $S_2 \subset B$ (so $|\text{Greedy}| = |S_1| = |S_2|$); then I named $A \setminus S_1$ as T_1 and $B \setminus S_2$ as T_2, and invited then to consider where the remaining edges of G might be. After some discussion among themselves, they agreed that there could be no edge in G that connects T_1 to T_2. I thought they would easily take the next step. I waited ... hoping that they would soon write down the crisp conclusion $|\text{Opt}| \leq |S_1| + |S_2| \leq 2 \cdot |\text{Greedy}|$. At this point everything I said started confusing them even more ... so I called *BREAK*, but the students seemed almost as frustrated as I was. They continued discussing why all my mysterious sounding claim were obvious after all!

During the break (with Ashwin's help) I tried to make some examples that better illustrated the point I made before the break. Some more got the idea; some others accepted what I said but didn't seem completely comfortable with the logic I was trying to push as *obvious*. I decided to move to the next exploration, but before that I defined *vertex cover* and wrote

$$|\mathsf{MaximumMatching}(G)| \_\_?\_\_ |\mathsf{MinimumVertexCover}(G)|,$$

and invited them to replace $\_\_?\_\_$ with an inequality; they said '$\leq$' was *obvious*—I felt reassured! I told them that for bipartite graphs the

opposite is true too:

(König-Egerváry theorem) In every bipartite graph

$$|\mathsf{MaximumMatching}| = |\mathsf{MinimumVertexCover}|.$$

I told them that we will prove this theorem by describing an algorithm for finding the maximum matching. Someone asked if it is easy to find $\mathsf{MinimumVertexCover}(G)$—I said, we will see about that. I began by asking them what could be *greedier than greedy.* I encouraged them to use the terminology of matched-vertices, free-vertices, matching edges, non-matching edges; they told me that we could grow the matching by adding two non-matching edges and remove one matching edge, or by adding three non-matching edges and removing two matching edges, or by adding four non-matching edges and removing three matching edges, and so on, ... I told them about *alternating paths* and *augmenting paths* and wrote down the algorithm (see fig. 7) that they essentially came up with. We use the following notation. For sets S and T, we write $S \oplus T$ for the set $(S \cup T) \setminus (S \cap T)$, consisting of the elements of that are in exacly one of the sets S and T. Note that if M is a matching, and P is an augmenting path in G with respect to M (both M and P are sets of edges), then $M' = M \oplus P$ is also a matching in G and $|M'| = |M| + 1$.

Step 0: Start with the empty matching, $M \leftarrow \emptyset$.

Step 1a: If G has an augmenting path P for improving M, then set $M \leftarrow M \oplus P$;

Step 1b: Else (G has no such augmenting path), output M and STOP.

Figure 7: The augmenting paths algorithm

Question: How are we sure that when the algorithm can find no more augmenting paths, the matching it currently has is optimal? I drew a picture (again S_1, S_2, T_1, T_2) and asked them to consider where the non-matching edges of G could be. I gave names to the set of vertices reachable by alternating paths from T_1 and T_2. They began to see a vertex cover of the same size as the matching, ..., clearly, when the algorithm stops it finds a matching of the maximum size. Finally, in a rush, I asked them to confirm that every r-regular $(r \geq 1)$ bipartite graph $G = (A, B, E)$ has a perfect matching? Suppose $|A|, |B| = n$. By Kőnig-Egerváry, it is enough to show that every vertex cover has at least n vertices. How many edges are there in G? How many of these edges does any one vertex cover? Someone asked if we could use induction to show that every r-regular bipartite graph has a perfect matching; someone mentioned Berge's theorem; someone asked if there was an efficient way to find augmenting paths; ..., *see you next time.*

Session II: 26 April 2025

Recap

See if you remember why the following inequalities hold; are they completely obvious?

$$|\mathsf{MaximumMatching}| \leq |\mathsf{MinimumVertexCover}| \leq 2 \cdot |\mathsf{Greedy}| \leq 2 \cdot |\mathsf{MaximumMatching}|.$$

The above inequalities hold for all graphs G. In the last session, we saw a proof of the Kőnig-Egerváry theorem which states that for bipartite graph the first inequality is actually an equality:

$$|\mathsf{MaximumMatching}| = |\mathsf{MinimumVertexCover}|.$$

We proved this miraculous equality using an algorithm (see fig. 7). Recall that given a matching M in a bipartite graph G, an alternating path in G (with respect to M) is a path where matching and non-matching edges alternate; an augmenting path is an alternating path that starts and ends at free vertices. Note that in fig. 8,

$$a_1 - - b_2 — a_3 - - b_5$$

is an alternating path (but it is not an augmenting path because b_5 is not free);

$$a_1 - - b_2 — a_3 - - b_4$$

is an augmenting path. For the algorithms to be effective, we need to describe a method for finding augmenting paths (someone asked me this last time as we were leaving).

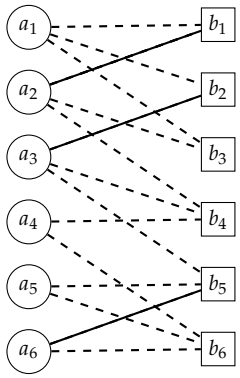

Figure 8: The matching edges are solid, the non-matching edges are dotted

The Hungarian exploration: The idea is to explore the graph $G = (A, B, E)$ in a breadth-first fashion, starting from the free vertices in A (see fig. 9). First, we assign directions to the edges of G. Direct all non-matching edges from A to B, and all matching edges from B to A. We will build a collection of trees by exploring this directed graph. Start with all free vertices in A, and write them on the top; these are the roots of our trees. Mark them *red.* If there is a directed (non-matching) edge (a, b) from a vertex a marked red to an unmarked vertex $b \in B$, then mark b *blue* and insert b as a child of a. If b is a free vertex, we have found an augmenting path; stop. Next, if there is a matching edge (b, a) from a vertex $b \in B$ marked blue to an unmarked vertex $a \in A$, then mark a *red*, and insert a as a child of b. Repeat until the trees cannot be grown anymore. If no augmenting path is found, then the current matching is already optimal!

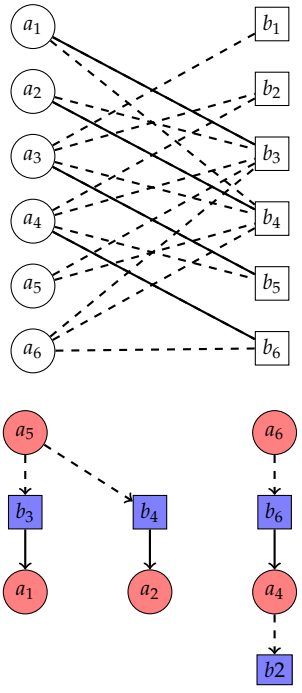

Figure 9: The Hungarian exploration of the graph above produces two trees

Suppose the Hungarian exploration does not yield an augmenting path. How will we find an a minimum vertex cover from the Hungarian tree?

Exploration IV: A perfect colouring

Partition the set of numbers $\{1, 2, \ldots, 18\}$ into six sets of 3 numbers each; call the six groups $S_1, S_2, S_3, S_4, S_5, S_6$ (the order in which the six groups are listed or the order in which the elements are listed in each group is not important). Once again partition $\{1, 2, \ldots, 18\}$ into six groups of 3 numbers each, completely arbitrarily; call the six group $T_1, T_2, \ldots, T_6$ (again the ordering does not matter). See fig. 10 for an example.

> We wish assign colours R, B, G to the numbers $1, 2, \ldots, 18$ with colours such that in each of the 12 groups, we have numbers of all three colours. Can this always be done? How will you phrase this as a bipartite matching problem? How many vertices? How many edges? Is your graph regular?

$S_1 = \{11, 2, 9\}$ $T_1 = \{1, 5, 8\}$
$S_2 = \{6, 18, 4\}$ $T_2 = \{15, 10, 3\}$
$S_3 = \{14, 5, 13\}$ $T_3 = \{4, 2, 7\}$
$S_4 = \{7, 16, 10\}$ $T_4 = \{13, 6, 17\}$
$S_5 = \{3, 17, 12\}$ $T_5 = \{18, 16, 9\}$
$S_6 = \{8, 15, 1\}$ $T_6 = \{12, 14, 11\}$

Figure 10: Color the numbers R, B, G so that every group receives numbers of all three colours

Exploration V: Guess the missing card

Disha, Sam and Supurna will show you a card trick. Sam will give you a pack of cards, Disha will stand near the board.

Step 1: You draw five cards from the deck of 52 cards.

Step 2: Sam will give you four of the cards one-by-one; each time you receive a card, you should write down its name on the board. The fifth card is kept separately.

Step 4: Disha will read what is on the board, and tell you the name of the 5th card. *Magic!*

How did they manage this? What is the bipartite graph $G = (A, B, E)$ that models this situation? Must this bipartite graph have a perfect matching for this trick to work? Does the graph always have a matching?

Optional exploration VI: Proofs using induction

Some of you attempted to use induction to prove that every r-regular $(r \geq 1)$ graph has a perfect matching. It is indeed possible to use induction to prove the theorem, but formulating the right induction hypothesis is tricky. Here are two induction-based approaches for statements involving matching in bipartite graphs.

Hall's theorem. Suppose in a bipartite graph $G = (A, B, E)$, for every subset $S \subseteq A$, we have $|N(S)| \geq |S|$. Then, G has a matching where every vertex in A is matched. (Do you see why this shows the statement about the existence of a matching in r-regular graphs?)

Complete the following induction-based proof. Fix a graph G satisfying the hypothesis, and assume the claim holds for all smaller graphs. Suppose $|N(S)| = |S|$ for a non-empty strict subset of A. Use induction to find a matching for S. Delete S and the vertices it is matched to from the graph. Argue that the hypothesis holds in the graph that remains, so by induction again, there is a matching for $A \setminus S$ in the graph that remains. What would you do if if there is no such subset S?

Another proof by induction: This proof directly establishes the Kőnig-Egerváry theorem by showing that.

$$|\mathsf{MaximumMatching}(G)| \geq |\mathsf{MinimumVertexCover}(G)|.$$

Please see Romeo Rizzi's paper in the Journal of Graph Theory.

Explorations on augmenting paths

Recall that we showed that the algorithm of fig. 7 produces a maximum cardinality matching when it stops by arguing that when no more augmenting paths exist, there is a vertex cover of the same cardinality as the current matching. Clearly, no matching can have cardinality greater than that of a vertex cover. So we indirectly established that if a matching is not of maximum cardinality, then there must be an augmenting path that improves it. This last statement can be shown more directly; what is more, it holds for all graphs, not just for bipartite graphs. Let us see why. Let $G = (V, E)$ be an undirected graph. Let $M \subseteq E$ be a matching in G. Suppose M' is another matching in G such that $|M'| > |M|$. Consider the edges in the set $M \oplus M'$; draw the edge of M as solid lines, and the edges of M' as wavy lines. What does each connected component of this graph look like? Will one of the components be an augmenting path with more wavy edges than solid edges? Someone mentioned Berge's theorem during the first session.

> *Berge's theorem.* Suppose M is a matching in an undirected graph $G = (V, E)$. If M is not of maximum cardinality, then there is an augmenting path that improves M, that is, an alternating path that starts and ends at free vertices.

Exploration VII: Oh stand, stand by me ...

Here is a game to be played by a Red player and a Blue player on a graph (e.g., fig. 11). The Red player chooses a vertex and marks it Red. Then, the Blue player must pick an unmarked vertex that is adjacent to the recently marked Red vertex and mark it Blue. Then,

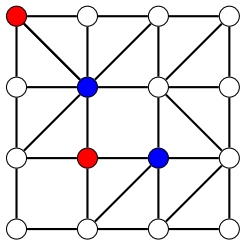

Figure 11: The players take turns to grow a path

the Red player must pick an unmarked neighbour of the recently marked Blue vertex and mark it Red. Then, the Blue player must ... and so on. A player looses if she has no unmarked vertex she can move to while respecting the rules of the game. Note that the game always ends.

Show that the Red player (who starts the game) has a winning strategy if and only if the graph has no perfect matching. (Hint: Berge's theorem has something to do with this, not hard!)

Optional exploration VIII: A factory with workers and tasks

Consider the example similar to the one we started with. Suppose $G = (A, B, E)$ models a factory, where A is the set of workers, and B is the set of tasks. An edge of the form (a, b) indicates that worker a is capable of performing task b. Now, on certain day the factory management proposes to get a set of tasks $B_1 \subseteq B$ completed; for this it proposes to use a matching M_1 where all tasks in B_1 are matched. On the other hand, the worker's union would like a set $A_1 \subseteq A$ of workers to be assigned tasks on that day, for which the union suggests a matching M_2 where all workers in A_1 are assigned tasks. Show the following (a result of Dulmage and Mendelsohn):

There is a matching M in G so that so that simultaneously all workers in A_1 are assigned tasks and all tasks in B_1 are assigned workers. (The factory management and the workers are both happy!)

Hint: Consider the alternating paths and cycles in $M_1 \oplus M_2$.

Report on the session of 26 April

We reviewed the basic graph-theoretic terminology the key inequalities connecting maximum matchings, the greedy solution, and vertex covers. The confusion in the last session seemed to have lifted without my saying anything (perhaps, because I said nothing). (On a parent's prompting, we tried to understand why the maximum matching and minimum cover are not necessarily of the same size in non-bipartite graphs: we drew cycle with five vertices.) We concluded that every r-regular bipartite graph has a perfect matching, and soon after that we concluded that bipartite graphs where all left vertices had degree r_1 and all right vertices had degree r_2 with $r_1 \geq r_2$ would also have a matching that saturates all vertices on the left. We discussed augmenting paths and the Hungarian exploration quickly, and moved on to Exploration IV. Luckily, 20 students showed up for the session, Disha partitioned them into 5 disjoint groups of four each in two ways: $(C_1, C_2, \ldots, C_5)$ and $(V_1, V_2, \ldots, V_5)$. The story was

that we wished to drive out for a picnic in cars, four to a car. However, we do not know whether the students will be distributed in the vehicles according to $(C_1, C_2, \ldots, C_5)$ or $(V_1, V_2, \ldots, V_5)$. Nevertheless, we needed to assign each student a cap in one of four colours beige, blue, pink and red, in such a way that no matter which scheme was used each vehicle has one student with each of the four colours. The modelling of this graph as a matching problem needs some thought. One is tempted to make vertices corresponding to colours and connect them somehow to the two sets of groups. It is not clear how the edges must be placed between the vertices to correctly capture the constraints we have. Several modelling the problem with bipartite graphs were suggested, but they seemed to fail to completely capture the constraints. Eventually, someone suggested that we build a bipartite graph with $(C_1, C_2, \ldots, C_5)$ or $(V_1, V_2, \ldots, V_5)$ as vertices and connect two groups with as many lines as they have students in common. Effectively, we had one edge for each student, connecting the two groups that the student belonged to. One quickly saw that this was a 4-regular bipartite graph. A matching was found in the graph; we went through the motions of a Hungarian exploration. Once the first matching was found, and the students whose edge formed this matching were given caps (pink was voted the most appropriate colour for them). Once these five students were out, we were left with a 3-regular graph, in which a matching was found again. Naturally, one wanted to continue, and use the theorem again ...

However, I pointed out that a 2-regular graph is just a collection of even cycles; finding a perfect matching in such graphs is straightforward, and we could just pick alternate edges in these cycles for a matching. Everybody put on their caps, and walked to their cars ... and went for a picnic.

After the break, I introduced the Exploration V. I wanted to explain the connection of the trick to matchings, but everybody wanted to see things for themselves and think about it. Disha, Sam and Supurna teamed up to execute the trick expertly; this generated much excitement. Then I drew the bipartite graph, with $\binom{52}{5}$ vertices on the left and $52 \cdot 51 \cdot 50 \cdot 49$ vertices on the right, corresponding to the number of possiblities for the five cards that are drawn from the pack, and the names of the four cards that are announced. Question: What is the left-degree r_1? They said: $5! = 120$. What is the right degree r_2? They said: 48. Since $120 > 48$, the graph must have a perfect matching that saturates the left-hand-side. So, in principle, there must be a way to get the trick done? What matching did Disha, Sam and Supurna use? What computations did they perform in their head? Now, we could consider a deck of n distinct cards instead of

a deck of 52 cards. Left degree r_1 is still 120; the right degree $n-4$. So the trick can be made to work as long as $n-4 < 120$ or $n \leq 124$. How does on do this in the head (see here)? I then briefly mentioned Exploration VII and left them to figure it out themselves!

Just asking ... multitasking?

Let us look at the first example again. Could we ask the workers to share the load and get more jobs done? Say, we ask w_2 to work on t_2 say for half the day, and w_3 to work on t_2 for half the day. Together, t_2 will receive the attention it needs, and will get done by the end of the day. Perhaps, we ask w_2 and w_3 to work on other tasks as well, when they are not working on task t_2. Similarly, we may ask other workers to work part-time on the jobs, making sure that no worker is assigned a total load of more than 1. We may think about this problem as follows. We wish to assign weights in the range $[0,1]$ to each edge so that

(i) The weights of the edges incident on any v_i add up to at most 1. (No worker is overloaded.)

(ii) The weights of the edges incident on any w_i add up to at most 1. (No task can be more than 100% done!)

We may then measure the total work done in the office as the sum of the weights of the edges. Now, if we insisted on workers being assigned to only one task for the day (which corresponds to whole number weights are 0 or 1), then clearly we cannot get more than four tasks done. Can assignments with general weights in $[0,1]$ improve the total work done in the office?

Fractional matching, fractional vertex covers: In a general bipartite graph $G = (A, B, E)$, a *fractional matching* is an assignment of non-negative weights f_e to the edges of G so that the weights assigned to the edges incident on a vertex v add up to at most 1. That is, suppose e_1, e_5, e_{11} are incident on v, then our assignment must be such that $f_{e_1} + f_{e_5} + f_{e_{11}} \leq 1$. Note we have combined the two conditions (i) and (ii) into one. We continue to use the word *size* to evaluate how big a given fractional matching is: the size of the matching is the sum the weights assigned to edges. Clearly, we wish to find a fractional matching of the largest size. We can also consider fractional covers. Imagine that a city, in order to fight fires, wishes to build water tanks at the various street intersections. The usual vertex cover, corresponds to building tanks of unit capacity, so that there is enough water in a single tank to control any fire in any of the streets

that arrive at the intersection. The minimum vertex cover then corresponds to the minimum total amount of water we need to store so that all streets are served. What if we are allowed to build tanks of various capacities? We only require that the total amount of water available at the two ends of every street is at least one unit. This is the factional vertex cover problem: assign capacities $c_v \in [0,1]$ to the vertices of G so that for every edge of the form (v,w) in the graph, we have $c_v + c_w \geq 1$. The *size* of such a vertex cover is the sum of the capacities assigned to the vertices. Can you show the following inequalities? Are the first and last obvious? The second is not hard either? Maybe, the third will take a little more effort.

$$|\text{Maximum matching}| \leq \min \text{size}(\text{fractional vertex cover}) \leq |\text{Minimum cover}|;$$
$$|\text{Maximum matching}| \leq \max \text{size}(\text{fractional matching}) \leq |\text{Minimum cover}|.$$

Go back to the Kőnig-Egerváry theorem. What does it now say about $\min \text{size}(\text{fractional vertex cover})$ and $\text{size}(\text{fractional matching})$. Surprised?

Some advanced stuff: We saw that in non-bipartite graphs the maximum size of matching can be strictly smaller than minimum size of a vertex cover (recall the pentagon). In particular, we do not have a Kőnig-Egerváry theorem any longer to compare the sizes of fractional matchings and fractional vertex covers. With a little work (ask me if you need a hint) one can show that for every fractional matching and fractional vertex cover:

$$\text{size}(\text{fractional matching}) \leq \text{size}(\text{fractional vertex cover}).$$

As we did before, we can stick in a max on the left and a min on the right.

Theorem 1. $\max \text{size}(\textit{fractional matching}) = \min \text{size}(\textit{fractional vertex cover})$.

So, in non-bipartite graphs, the maximum size of matching maybe strictly smaller than the size of the minimum vertex cover; nevertheless, the fractional versions of these quantities. No doubt this is beautiful, and leads to many wonderful things ... which we can perhaps discuss some other time.

..., for now, however, we must return to bipartite graphs and ...

Go with the flow ...

Let us return to fractional matchings in bipartite graphs. How does one visualize a fractional matching of the largest size? Look at the network in fig. 12. Our bipartite graph is embedded in the middle

of this network. We have attached a source vertex s and a terminal vertex t (also called sink vertex). On the directed edges, we place capacities (in our case, all capacities are 0, 1 or ∞, but later they will be other numbers).

Imagine that we wish to send flows along the edges so that (i) at all vertices (except s and t) the total incoming flow is equal to the total outgoing flow; (ii) the flow on no edge exceeds its capacity. Such an assignment of flows to the edges is said to be *valid*. The value of the flow is net flow leaving s. Do you see that the size of the maximum fractional matching in the bipartite graph is exactly the maximum value of a flow in the network of fig. 12.

In the next two session, we will study flows in networks.

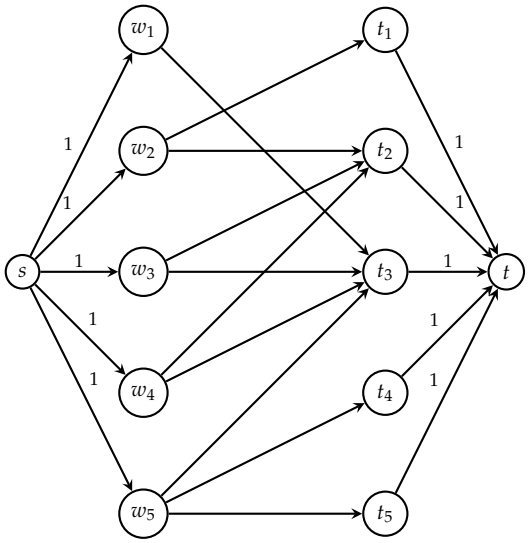

Figure 12: The network-flow graph for the office (the edges in the middle have infinite capacity)