Outline

In Lecture 1, we first learn the basics of graph theory that consistutes the fundamental ground of combinatorial optimization. Then we get to introduce one of our main topics for this course, the bipartite matching problem. We also analyze the greedy algorithm that finds a maximal matching.

1 Graph theory basics

Königsberg is the historic German and Prussian name of the medieval city that is now Kaliningrad, Russia. Figure 1.1 shows a 18th-century map of the city, highlighting two islands in the middle surrounded by the river and the seven bridges. Their citizens were interested in the following

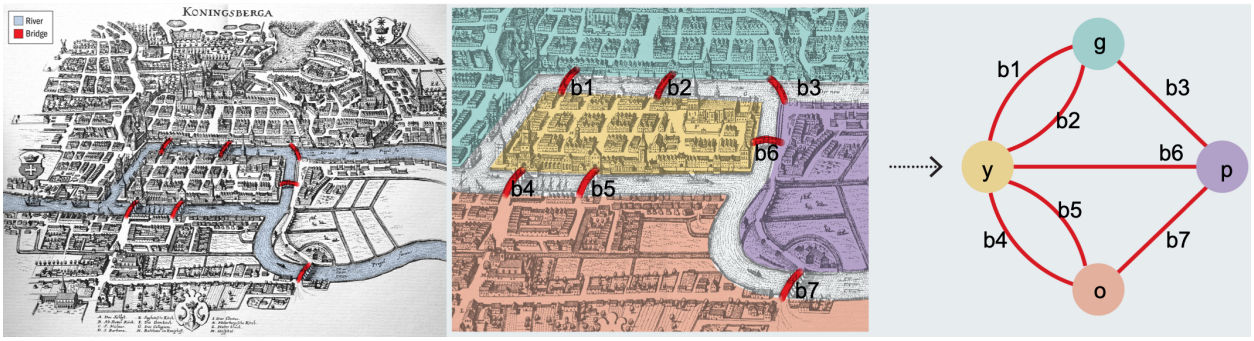

Figure 1.1: the medieval city of Königsberg, Prussia and the seven bridge problem

question, that is now known as the famous Königsberg bridge problem. Starting from a certain region, is it possible to traverse all bridges exactly once and come back to the initial location? To answer this question, one can abstract out the map, focusing on the connections between the four different regions by the seven bridges. The second part of Figure 1.1 colors the four regions green, yellow, purple, and orange, respectively. For simplicity, we refer to them as g, y, p, and o. We enumerate the seven bridges as $b_1, \dots, b_7$. Here, bridges b_1 and b_2 connect g and y. Other bridges connect two different regions, crossing the river. We can represent this information as in the third part of Figure 1.1. Once we have simplified the picture, it becomes easier to argue that the answer to the question is no. The argument is as follows. When we leave a region and come back to it later, we take two bridges attached to the region. Hence, returning to the initial location would take an even number of bridges connected to it. However, as shown in Figure 1.1, each region is attached to an odd number of bridges.

Elements of graphs The third picture of Figure 1.1 is referred to as a **graph**. A graph G consists of its **vertex** set V and an **edge** set E. For example, a vertex is used to represent a region in a map as in Figure 1.1 while an edge connects two vertices like a bridge. That said, the Königsberg map can be represented by a graph G whose vertex set is given by $V = \{g, y, p, o\}$ with the edge set $E = \{b_1, \dots, b_7\}$. Given two vertices $u, v \in V$, an edge connecting u and v can be expressed as the set notation $\{u, v\}$ or simply as uv. Here, we say that two vertices u and v are **adjacent** if there is an edge $uv \in E$ between them. Moreover, we call any vertex adjacent to u a

neighbor of u. Next, given an edge $e = \{u, v\}$, we call u and v the **endpoints** of e. We say that a vertex u is **incident** to an edge e if u is one of the endpoints of e. Lastly, the **degree** of a vertex is the number of its neighbors which equals the number of incident edges to it.

Complete and bipartite graphs Let us consider several examples of graphs. We have four

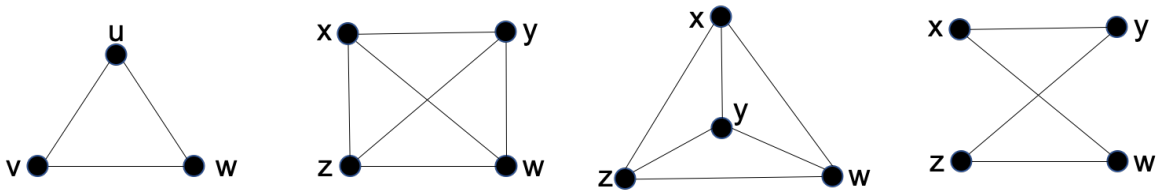

Figure 1.2: examples of graphs

graphs in Figure 1.2. The first one has the vertex set $V = \{u, v, w\}$ and the edge set $\{uv, vw, wu\}$. The second one has the vertex set $\{x, y, z, w\}$ and the edge set $\{xy, xw, xz, yz, yw, zw\}$. In fact, the third graph is **isomorphic** to the second one. Note that the first graph has an edge between every pair of two vertices, which is also the case for the second example. Such a graph is called a **complete** graph. A complete graph on n vertices is denoted by K_n. In a complete graph, any two distinct vertices are adjacent, and K_n has $\binom{n}{2}$ edges. The fourth example has the same vertex set as the second and third ones while the edge set is given by $\{xy, xw, yz, zw\}$. There is no edge between x and z and no edge between y and w. This graph is a **bipartite** graph. In general, a graph is bipartite if its vertex set V can be partitioned into two disjoint sets V_1, V_2 and no two vertices inside each of V_1 and V_2 are not adjacent.In the fourth example of Figure 1.2, the vertex set is partitioned into $\{x, z\}$ and $\{y, w\}$. A **complete bipartite** graph is a bipartite graph with the vertex set partition $V = V_1 \cup V_2$ such that there is an edge between any $u \in V_1$ and $v \in V_2$. We use notation $K_{m,n}$ for a complete bipartite graph where one vertex subset has m vertices and the other has n vertices. The fourth example in Figure 1.2 is indeed a complete bipartite graph $K_{2,2}$. Figure 1.3 shows bipartite graphs whose vertex set can be partitioned into two subsets of

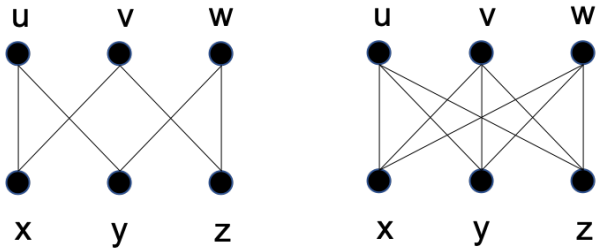

Figure 1.3: a bipartite graph(left) and a complete bipartite graph(right)

three vertices. The second graph of Figure 1.3 is $K_{3,3}$.

Paths and cycles A **walk** is a sequence of vertices $v_0, v_1, \ldots, v_k$ where any two consecutive vertices are adjacent, i.e., $v_{i-1}v_i \in E$ for any $i \in \{1, \ldots, k\}$. An uv**-walk** is a walk whose initial vertex is u and the final vertex is v. A **path** is a walk where no vertex is visited more than once. An uv**-path** is a path that starts with u and ends with v. Figure 1.4 shows a walk and a path. A **closed walk** is a walk $v_0, v_1, \ldots, v_k$ that ends with the initial vertex, i.e., $v_0 = v_k$. A **cycle** is a

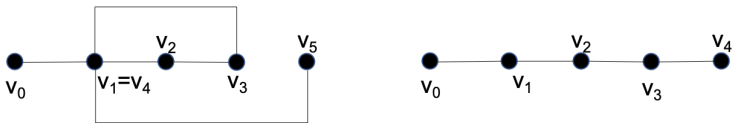

Figure 1.4: a walk(left) and a path(right)

closed walk $v_0, \ldots, v_k$ that repeats no vertex except for $v_0 = v_k$. Figure 1.5 shows a closed walk and a cycle. The **length** of a walk is the number of edges that it takes. A **shortest path** between

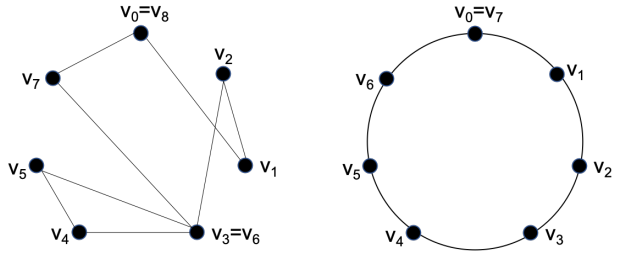

Figure 1.5: a closed walk(left) and a cycle(right)

two vertices u and v is an uv-path with the minimum length. A cycle of an odd length is referred to as an **odd** cycle, and a cycle of an even length is called an **even** cycle.

Subgraphs and cliques Given a graph $G = (V, E)$, a **subgraph** of G is given by $G' = (V', E')$ such that $V' \subseteq V$, $E' \subseteq E$, and E' is defined over V', which means that the endpoints of the edges in E' are fully contained in V'. An **induced subgraph** H of G is a subgraph of G defined with a vertex subset $U \subseteq V$ so that uv is an edge in H if and only if $u, v \in U$ and $uv \in E$. Here, we say that H is a **subgraph induced by** U. In Figure 1.6, the graph in the middle is a subgraph of the first graph. However, it is not an induced subgraph as the edge v_2v_4 is missing. The third

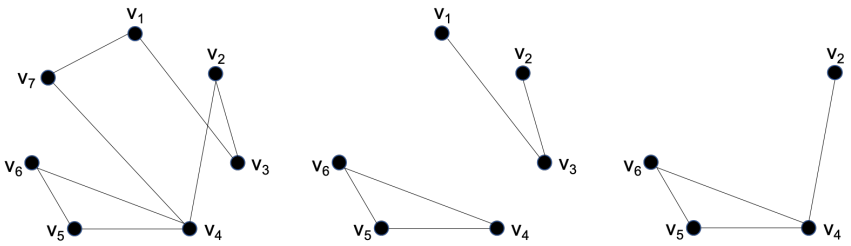

Figure 1.6: a graph, a subgraph, and an induced subgraph

graph is indeed an induced subgraph, which is given by the vertex subset $\{v_2, v_4, v_5, v_6\}$. A **clique** is a subgraph that is a complete graph. In Figure 1.6, the subgraph induced by the vertex subset $\{v_4, v_5, v_6\}$ is K_3, so it is a clique.

Based on the concepts that we have learned so far, let us establish another equivalent characterization of bipartite graphs. First, the following statement will be useful.

Lemma 1.1. *A closed walk of an odd length contains an odd cycle as a subgraph.*

We leave the lemma as an exercise. With this, we are ready to prove the following characterization of bipartite graphs.

Proposition 1.2. *A graph is bipartite if and only if it has no odd cycle as a subgraph.*

Proof. Let us first argue that a biparite graph G contains no odd cycle. Suppose for a contradiction that G has an odd cycle C, given by $v_0, v_1, \ldots, v_k$ with $v_0 = v_k$ where k is odd. Let the vertex set of G be partitioned into V_1 and V_2. By definition, any two consecutive vertices in C belong to distinct subsets. Without loss of generality, assume that $v_0 \in V_2$. As v_0v_1 is an edge, $v_1 \in V_1$. Repeating the same argument for the other edges of C, as $v_{k-1}v_k$ is an edge and k is odd, $v_k \in V_1$. This contradicts the assumption that $v_0 = v_k$.

Next we show that if a graph G has no odd cycle, then it is bipartite. Take a vertex u of G. Then define V_1 as the set of vertices, to which a shortest path from u has an even length. Similarly, let V_2 be defined as the set of vertices, to which a shortest path from u has an odd length. As 0 is even, u itself belongs to V_1. By definition, V_1 and V_2 provide a proper partition of the vertex set. Suppose for a contradiction that there exist two distinct vertices v and w that are adjacent and belong to the same side of the partition. Hence, vw is an edge, and $v, w \in V_1$ or $v, w \in V_2$. Then let us construct a closed walk as follows. We first take a shortest path from u to v. Then we move from v to w by the edge vw. Lastly, we take a shortest path from w to u. Since v and w belong to the same side of the partition, the uv-shortest path and the wu-shortest path have the same parity. This means that the closed walk has an odd length. Then, by Lemma 1.1, the closed walk contains an odd cycle, contradicting the assumption that G has no odd cycle. □

2 Bipartite matching problem

Recall that a bipartite graph is a graph $G = (V, E)$ where

- the vertex set V is partitioned into two sets V_1 and V_2,
- each edge $e \in E$ crosses the partition, i.e. e has one end in V_1 and the other end in V_2.

For example, Figure 1.7 shows a bipartite graph on four vertices where the vertex set is partitioned into two sets of two vertices. A **matching** is a set of edges without common vertices. In Figure 1.7,

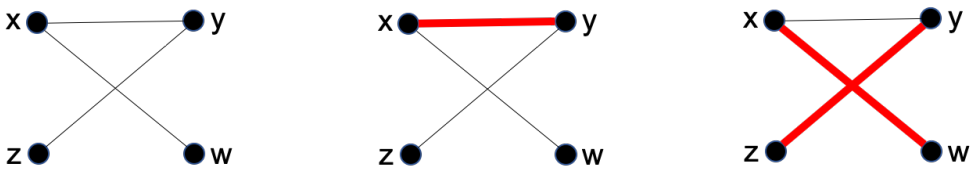

Figure 1.7: a bipartite graph and its matchings

the second and third figures provide matchings of the bipartite graph. A **maximal matching** is a matching that cannot be extended to a matching of a larger cardinality. In other word, a maximal matching is not properly contained in another matching. Note that the second figure of Figure 1.7 is a maximal matching. A **maximum matching** is a matching with the maximum number of edges. The third figure of Figure 1.7 is a maximum matching. Hence, a maximum matching is a maximal matching, while the converse is not true.

The **maximum bipartite matching** problem is to find a matching that has the maximum number of edges. The maximum bipartite matching problem can model a wide range of applications in practice. As the name suggests, it provides a natural framework for couple matching scenarios. Similarly, US medical students are matched with residency programs using the National Resident

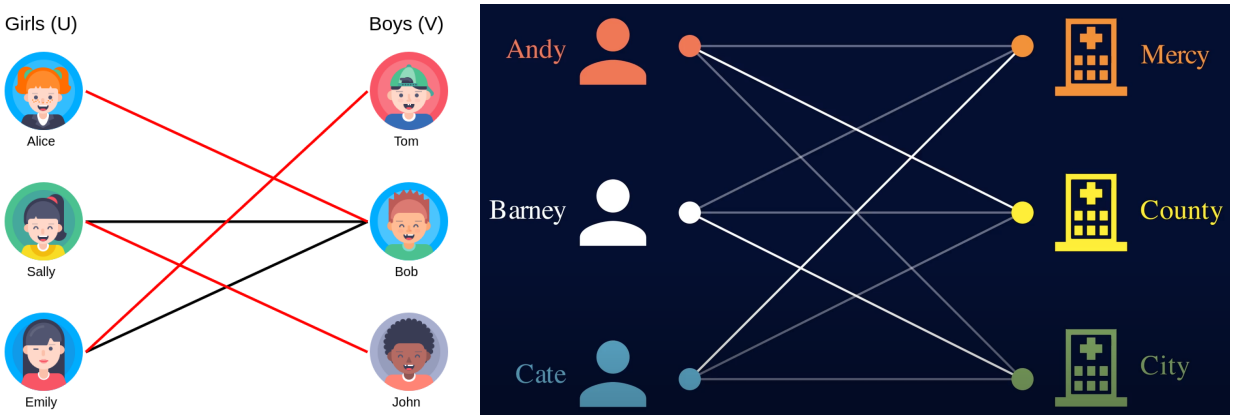

Figure 1.8: couple matching and doctor-hospital assignment applications

Matching Program (NRMP) based on bipartite matching, as described in Figure 1.8. Online advertisement slot allocation is another exemplary application of bipartite matching. Online platforms run auctions to sell their advertisement slots to advertisers, as shown in Figure 1.9. There are many

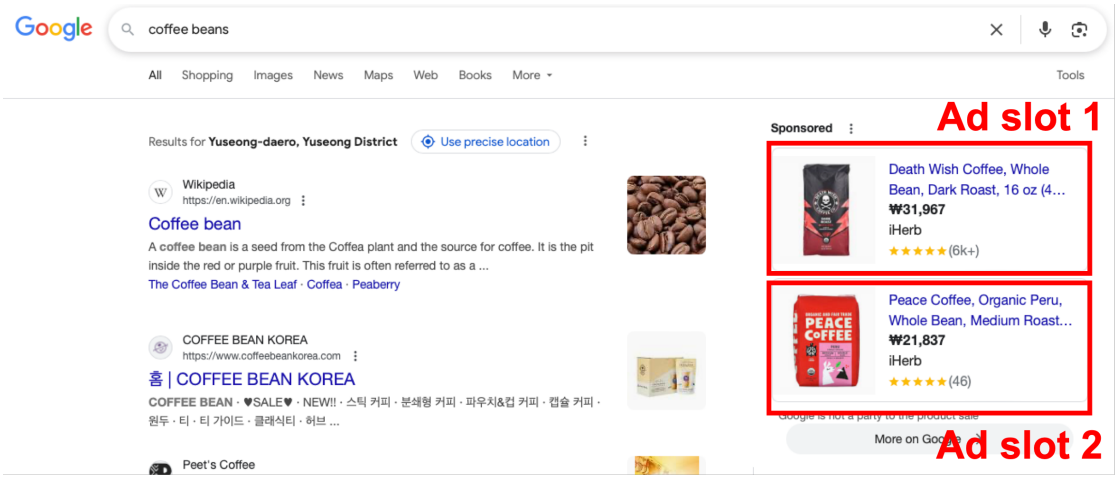

Figure 1.9: online advertisement slot allocation

other applications, and we list a few below.

- Recommender systems: A bipartite graph can represent user-item preference information.
- Economic matching markets: Bipartite networks can model how markets work between two disjoint parties of players, such as buyers and sellers. Online ad allocation is an example.
- Job-server scheduling: In a large data center, jobs to be processed arrive in real time, and the scheduler assigns them to multiple servers and processors based on their computing requirements.

3 Greedy algorithm for maximum bipartite matching

Now that we have introduced bipartite matching, our next question is about how to compute a maximum matching in a bipartite graph. One natural approach is to **greedily** select and add edges without destroying the matching structure. The greedy algorithm works as follows.

Algorithm 1 Greedy algorithm for bipartite matching

Enumerate the edges in E as $e_1, \ldots, e_{|M|}$
Initialize $M = \emptyset$.
for $i = 1, \ldots, |E|$ **do**
    If $M \cup \{e_i\}$ is a matching, then update M as $M = M \cup \{e_i\}$.
end for
Return M

Does the greedy algorithm given in Algorithm 1 find a maximum matching for a bipartite graph? Unfortunately, that is not always the case. In Figure 1.7, we have a bipartite graph with $V = \{x, y, z, w\}$ and $E = \{xy, xw.yz\}$. If the greedy algorithm processes the sequence xy, xw, yz, then it would return $M = \{xy\}$, whereas the maximum matching is given by $\{xw, yz\}$.

Nevertheless, can we understand how small a matching returned by the greedy algorithm is compared to the size of a maximum matching? The following result shows that the greedy algorithm can achieve at least 50% of the maximum size.

Theorem 1.3. *Let $G = (V, E)$ be a bipartite graph, and let M be a maximal matching. Then*

$$|M| \geq \frac{1}{2}\text{OPT}$$

where OPT *denotes the maximum size of a matching in G.*

Proof. We prove that for any matching M' in G, $2|M| \geq |M'|$. Suppose for a contradiction that $2|M| < |M'|$ for some matching M'. Let $e = \{u, v\}$ be an edge contained in M. Note that M' contains at most two edges that intersect with e. Since $2|M| < |M'|$, M' has an edge e' that intersects with none of the edges of M. This implies that $M \cup \{e'\}$ is a matching, contradicting the maximality of M. □

Since Algorithm 1 always finds a maximal matching, it follows that the number of edges in the matching M found by Algorithm 1 is at least half of the maximum size of a matching in a bipartite graph G. Therefore, the greedy algorithm is a $(1/2)$-**approximation algorithm** for maximum bipartite matching.