Lecture 1: introduction to bipartite matching

Dabeen Lee

Industrial and Systems Engineering, KAIST

2025 Winter Lecture Series on Combinatorial Optimization

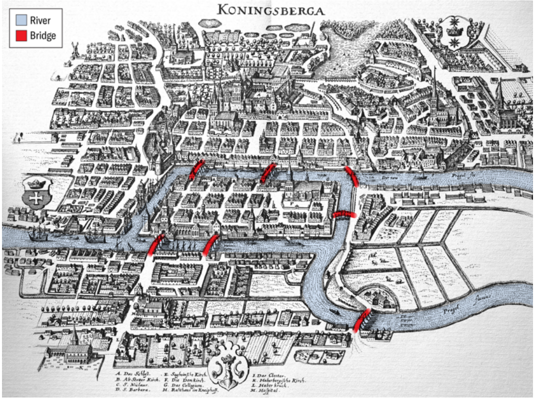
January 13, 2025

Outline

- Graph theory basics
- Bipartite matching problem
- Greedy algorithm for maximum bipartite matching

Introduction

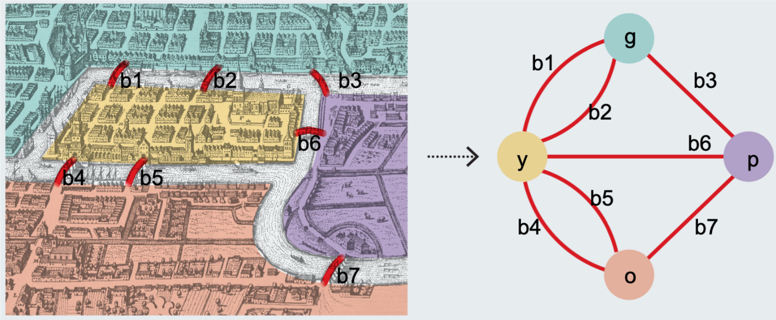
- Königsberg



- Question: Starting from a certain region, is it possible to traverse all bridges exactly once and come back to the initial location?

Introduction

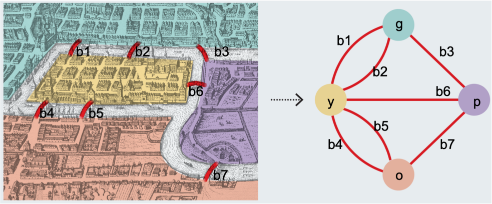
- Abstraction of Königsberg's map



- The second figure is referred to as a **graph**.

Elements of graphs

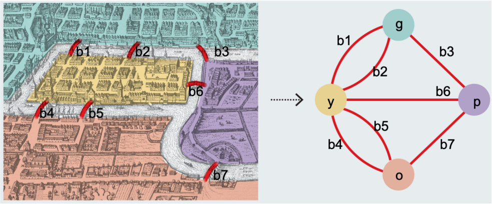
- A graph G consists of its **vertex** set V and an **edge** set E.



- 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.
- We say that a vertex u is **incident** to an edge e if u is one of the endpoints of e.
- The **degree** of a vertex is the number of its neighbors which equals the number of incident edges to it.

Elements of graphs

- A graph G consists of its **vertex** set V and an **edge** set E.



- 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.
- We say that a vertex u is **incident** to an edge e if u is one of the endpoints of e.
- The **degree** of a vertex is the number of its neighbors which equals the number of incident edges to it.

Complete graphs

Bipartite graphs

Complete bipartite graphs

Paths

- 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 $i \in \{1, \ldots, k\}$.

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

- 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.

Cycles

- 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 closed walk $v_0, \ldots, v_k$ that repeats no vertex except for $v_0 = v_k$.

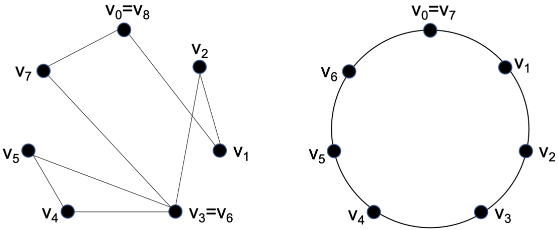


Figure: a closed walk(left) and a cycle(right)

Length and parity

- The **length** of a walk is the number of edges that it takes.
- A **shortest path** between 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.
- A cycle of an even length is called an **even** cycle.

Subgraphs

- 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'.
- This means that the endpoints of the edges in E' are fully contained in V'.

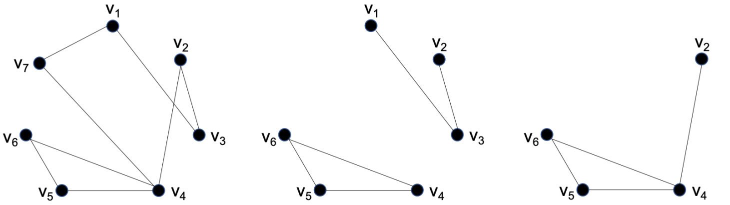


Figure: a graph, a subgraph, and an induced subgraph

- 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 $u, v \in U$.
- Here, we say that H is a **subgraph induced by** U.
- A **clique** is a subgraph that is a complete graph.

Equivalent characterization of bipartite graphs

Lemma

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

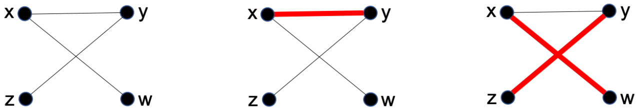
Proposition

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

Equivalent characterization of bipartite graphs

Bipartite matching

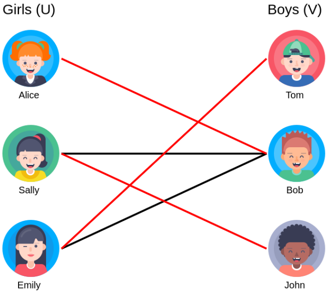
- Recall that a bipartite graph is a graph $G = (V, E)$ where
  - the vertex set V is partitioned into two sets V_1 and V_2,
  - each edge $e \in E$ crosses the partition, i.e. e has one end in V_1 and the other end in V_2.



- A **matching** is a set of edges without common vertices.
- 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.
- A **maximum matching** is a matching with the maximum number of edges.

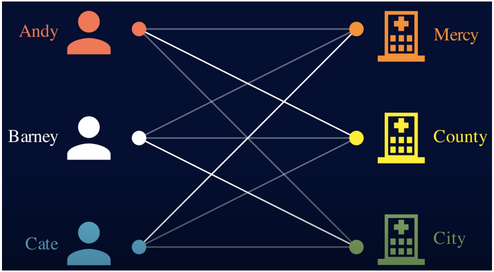
Maximum bipartite matching problem

- 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.



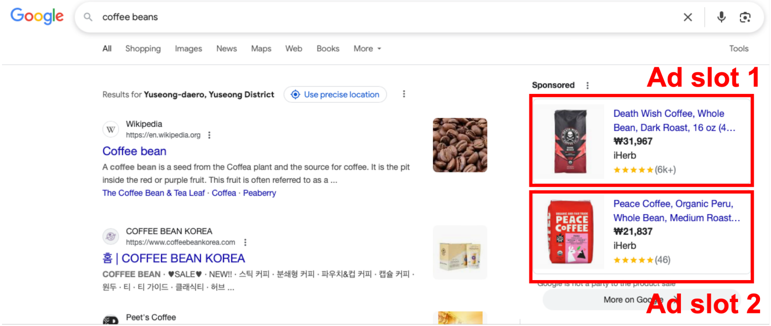
Applications of bipartite matching

- US medical students are matched with residency programs by the National Resident Matching Program (NRMP) based on bipartite matching.



Applications of bipartite matching

- Online advertisement slot allocation is another exemplary application of bipartite matching.
- Online platforms run auctions to sell their advertisement slots to advertisers.



Applications of bipartite matching

- **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.

Greedy algorithm for maximum bipartite matching

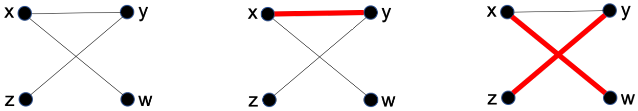
- How do we compute a maximum matching in a bipartite graph?
- One natural approach is to **greedily** select and add edges without destroying the matching structure.

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

Greedy algorithm for maximum bipartite matching

- Does the greedy algorithm find a maximum matching for a bipartite graph?
- Unfortunately, that is not always the case.



Greedy algorithm for maximum bipartite matching

- 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

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

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

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

- Since the greedy algorithm always finds a maximal matching, it is an $(1/2)$-**approximation algorithm** for maximum bipartite matching.

Greedy algorithm for maximum bipartite matching