Finding shortest paths

Definition

Given vertices u, v in weighted graph, the distance $d(u, v)$ is the minimum sum of the weights on the edges of a u, v-path.

Finding shortest paths

Definition

Given vertices u, v in weighted graph, the distance $d(u, v)$ is the minimum sum of the weights on the edges of a u, v-path.

We will describe an algorithm that, given a weighted graph and a vertex u, it finds the shortest path to every vertex.

It is called **Dijkstra's algorithm**.

Finding shortest paths

Definition

Given vertices u, v in weighted graph, the distance $d(u, v)$ is the minimum sum of the weights on the edges of a u, v-path.

We will describe an algorithm that, given a weighted graph and a vertex u, it finds the shortest path to every vertex.

It is called **Dijkstra's algorithm**.

We denote the weight of the edge $xy \in E(G)$ by $w(xy)$.
We set $w(xy) = \infty$ if $xy \notin E(G)$.

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u, v-path is known.

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u, v-path is known.
- A function $t(z)$ which records the *tentative* distance from u to z, using only vertices in S.

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u, v-path is known.
- A function $t(z)$ which records the *tentative* distance from u to z, using only vertices in S.

Initialization:

- $S = \{u\}$

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u, v-path is known.
- A function $t(z)$ which records the *tentative* distance from u to z, using only vertices in S.

Initialization:

- $S = \{u\}$
- $t(u) = 0, \;\; t(z) = w(uz)$ if $uz \in E(G), \;\; t(z) = \infty$ otherwise

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u, v-path is known.
- A function $t(z)$ which records the *tentative* distance from u to z, using only vertices in S.

Initialization:

- $S = \{u\}$
- $t(u) = 0, \;\; t(z) = w(uz)$ if $uz \in E(G), \;\; t(z) = \infty$ otherwise

Iteration:

- Choose $v \notin S$ such that $t(v) = \min_{z \notin S} t(z)$

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u, v-path is known.
- A function $t(z)$ which records the *tentative* distance from u to z, using only vertices in S.

Initialization:

- $S = \{u\}$
- $t(u) = 0, \;\; t(z) = w(uz)$ if $uz \in E(G), \;\; t(z) = \infty$ otherwise

Iteration:

- Choose $v \notin S$ such that $t(v) = \min_{z \notin S} t(z)$
- Add v to S.

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u,v-path is known.
- A function $t(z)$ which records the *tentative* distance from u to z, using only vertices in S.

Initialization:

- $S = \{u\}$
- $t(u) = 0, \;\; t(z) = w(uz)$ if $uz \in E(G), \;\; t(z) = \infty$ otherwise

Iteration:

- Choose $v \notin S$ such that $t(v) = \min_{z \notin S} t(z)$
- Add v to S.
- For each edge vz with $z \notin S$, update $t(z)$ to $\min\{t(z), t(v) + w(vz)\}$.

Dijkstra's algorithm

Input: A weighted graph G and a vertex u.

At any time, we maintain:

- A set S of vertices $v \in V(G)$ for which the shortest u, v-path is known.
- A function $t(z)$ which records the *tentative* distance from u to z, using only vertices in S.

Initialization:

- $S = \{u\}$
- $t(u) = 0, \;\; t(z) = w(uz)$ if $uz \in E(G), \;\; t(z) = \infty$ otherwise

Iteration:

- Choose $v \notin S$ such that $t(v) = \min_{z \notin S} t(z)$
- Add v to S.
- For each edge vz with $z \notin S$, update $t(z)$ to $\min\{t(z), t(v) + w(vz)\}$.

End when $S = V(G)$. Set $d(u, v) = t(v)$ for all v.

Dijkstra's algorithm

[Example]

Dijkstra's algorithm

[Example]

Theorem

Dijkstra's algorithm computes $d(u, z)$ for every $z \in V(G)$.

Dijkstra's algorithm

[Example]

Theorem

Dijkstra's algorithm computes $d(u,z)$ for every $z \in V(G)$.

In addition, one can reconstruct the shortest paths by recording, for each z, which is the chosen vertex v when $t(z)$ is updated. This means that the shortest u,z-path ends with the edge vz.

Dijkstra's algorithm

[Example]

Theorem

Dijkstra's algorithm computes $d(u, z)$ for every $z \in V(G)$.

In addition, one can reconstruct the shortest paths by recording, for each z, which is the chosen vertex v when $t(z)$ is updated. This means that the shortest u, z-path ends with the edge vz.

The same algorithm also works for digraphs.

Dijkstra's algorithm

[Example]

Theorem

Dijkstra's algorithm computes $d(u, z)$ for every $z \in V(G)$.

In addition, one can reconstruct the shortest paths by recording, for each z, which is the chosen vertex v when $t(z)$ is updated. This means that the shortest u, z-path ends with the edge vz.

The same algorithm also works for digraphs.

The special case of unweighted graphs is called **Breadth First Search**.

Chapter 3
Matchings

Matchings

Example 1: After medical school, students become residents at hospitals. Assigning students to hospitals is a complex problem as there are many factors to take into consideration. Consider a simplified version where

- each student is willing to go to some hospitals and the choices are not ranked,
- each hospital will accept at most one student.

Matchings

Example 1: After medical school, students become residents at hospitals. Assigning students to hospitals is a complex problem as there are many factors to take into consideration. Consider a simplified version where

- each student is willing to go to some hospitals and the choices are not ranked,
- each hospital will accept at most one student.

Example 2: The housing office has to distribute $2n$ students into n rooms (two in each room). Some pairs are compatible as roommates, some aren't. Under what conditions can they be all paired up?

Matchings

Example 1: After medical school, students become residents at hospitals. Assigning students to hospitals is a complex problem as there are many factors to take into consideration. Consider a simplified version where

- each student is willing to go to some hospitals and the choices are not ranked,
- each hospital will accept at most one student.

Example 2: The housing office has to distribute $2n$ students into n rooms (two in each room). Some pairs are compatible as roommates, some aren't. Under what conditions can they be all paired up?

These problems can be modeled in terms of finding matchings in graphs.

Matchings

Definition

A **matching** M in a graph G is a set of edges with no shared endpoints.

Matchings

Definition

A **matching** M in a graph G is a set of edges with no shared endpoints.

We say that a vertex is **saturated** if it is incident to some edge in M, otherwise it is called **unsaturated**.

Matchings

Definition

A **matching** M in a graph G is a set of edges with no shared endpoints.

We say that a vertex is **saturated** if it is incident to some edge in M, otherwise it is called **unsaturated**.

Definition

A **perfect matching** is one that saturates every vertex.

Counting matchings

How many perfect matchings does $K_{n,n}$ have?

Counting matchings

How many perfect matchings does $K_{n,n}$ have? $n!$

Counting matchings

How many perfect matchings does $K_{n,n}$ have? $n!$

How many perfect matchings does K_{2n} have?

Counting matchings

How many perfect matchings does $K_{n,n}$ have? $n!$

How many perfect matchings does K_{2n} have?

$$(2n-1)\cdot(2n-3)\cdot\cdots\cdot 3\cdot 1 = \frac{(2n)!}{2^n n!}$$

Counting matchings

How many perfect matchings does $K_{n,n}$ have? $n!$

How many perfect matchings does K_{2n} have?

$$(2n-1)\cdot(2n-3)\cdot\dots\cdot 3\cdot 1 = \frac{(2n)!}{2^n n!}$$

Does every connected graph with an even number of vertices have a perfect matching?

Counting matchings

How many perfect matchings does $K_{n,n}$ have? $n!$

How many perfect matchings does K_{2n} have?

$$(2n-1)\cdot(2n-3)\cdot\dots\cdot3\cdot1=\frac{(2n)!}{2^n n!}$$

Does every connected graph with an even number of vertices have a perfect matching? No.

Maximal and maximum matchings

Definition

A **maximal matching** is one that cannot be enlarged by adding more edges to it.

Maximal and maximum matchings

Definition

A **maximal matching** is one that cannot be enlarged by adding more edges to it.
A **maximum matching** is one of maximum size among all matchings in the graph.

Maximal and maximum matchings

Definition

A **maximal matching** is one that cannot be enlarged by adding more edges to it.
A **maximum matching** is one of maximum size among all matchings in the graph.

Maximum implies maximal, but not the other way.

Maximal and maximum matchings

Definition

A **maximal matching** is one that cannot be enlarged by adding more edges to it.
A **maximum matching** is one of maximum size among all matchings in the graph.

Maximum implies maximal, but not the other way.

Definition

Given a graph G and a matching M, a path in G is called **M-alternating** if its edges alternate between edges in M and edges not in M.

Maximal and maximum matchings

Definition

A **maximal matching** is one that cannot be enlarged by adding more edges to it.
A **maximum matching** is one of maximum size among all matchings in the graph.

Maximum implies maximal, but not the other way.

Definition

Given a graph G and a matching M, a path in G is called **M-alternating** if its edges alternate between edges in M and edges not in M.
If, additionally, its endpoints are unsaturated by M, the path is called **M-augmenting**.

Maximal and maximum matchings

Definition

A **maximal matching** is one that cannot be enlarged by adding more edges to it.
A **maximum matching** is one of maximum size among all matchings in the graph.

Maximum implies maximal, but not the other way.

Definition

Given a graph G and a matching M, a path in G is called **M-alternating** if its edges alternate between edges in M and edges not in M.
If, additionally, its endpoints are unsaturated by M, the path is called **M-augmenting**.

Note: If M is maximum, there is no M-augmenting path. We will show that the converse is also true.

Comparing matchings

Definition

The **symmetric difference** of M and M' consists of those edges that appear in exactly one of M and M':

$$M \triangle M' = (M \setminus M') \cup (M' \setminus M).$$

Comparing matchings

Definition

The **symmetric difference** of M and M' consists of those edges that appear in exactly one of M and M':

$$M \triangle M' = (M \setminus M') \cup (M' \setminus M).$$

Lemma

If M and M' are matchings, then every component of $M \triangle M'$ is a path or an even cycle.

Comparing matchings

Definition

The **symmetric difference** of M and M' consists of those edges that appear in exactly one of M and M':

$$M \triangle M' = (M \setminus M') \cup (M' \setminus M).$$

Lemma

If M and M' are matchings, then every component of $M \triangle M'$ is a path or an even cycle.

Theorem (Berge '57)

A matching M in a graph G is a maximum matching if and only if G has no M-augmenting path.