Math 2603 - Lecture 21
Section 10.4 Shortest path algorithms

Bo Lin

November 7th, 2019

Weighted graphs

Motivation

In practice, sometimes we not only care about whether two vertices are connected, but also care about the **length** of the edge.

Motivation

In practice, sometimes we not only care about whether two vertices are connected, but also care about the **length** of the edge.

Remark

The length could represent different quantities, for example:

- *Distance between two cities;*
- *Time duration of flights;*
- *Delivery cost between two places;*
- *Response time between two nodes of internet.*

Then we have an abstraction of these quantities.

Weighted graphs

Definition

A **weighted graph** *is a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *together with a function* $w : \mathcal{E} \to [0, \infty)]$. *For an edge* e, *the nonnegative real number* $w(e)$ *is called the* **weight** *of* e. *The* **weight of a subgraph** *of* $\mathcal{G}$ *is the sum of the weights of the edges of the subgraph.*

Weighted graphs

Definition

A **weighted graph** *is a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *together with a function* $w : \mathcal{E} \to [0, \infty)]$. *For an edge* e, *the nonnegative real number* $w(e)$ *is called the* **weight** *of* e. *The* **weight of a subgraph** *of* $\mathcal{G}$ *is the sum of the weights of the edges of the subgraph.*

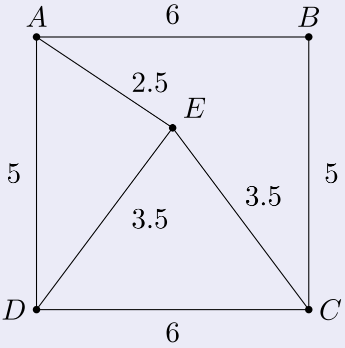
Remark

In most cases, the weights are positive. But sometimes we also have zero weight. Some people write ∞ *weight on non-existent edges.*

Example

Example

This is a weighted graph with 5 *vertices and* 7 *weighted edges.*



Distance problems

Traveling Salesman's Problem

Remark

On a business trip, a traveling salesman visits various towns and cities. If he wants to avoid having to pass through the same community twice, he needs a Hamiltonian cycle. In addition, there is a distance/mileage between any two adjacent communities, so he also wants to minimize the total distance he covers.

Traveling Salesman's Problem

Remark

On a business trip, a traveling salesman visits various towns and cities. If he wants to avoid having to pass through the same community twice, he needs a Hamiltonian cycle. In addition, there is a distance/mileage between any two adjacent communities, so he also wants to minimize the total distance he covers.

Definition

The **Traveling Salesman's Problem** *(TSP for short) is to find a Hamiltonian cycle with minimal weight in a weighted graph.*

It is hard

Remark

As we explained last time, finding a Hamiltonian cycle of a graph is already a hard problem, let alone finding one with minimal weight.

It is hard

Remark

As we explained last time, finding a Hamiltonian cycle of a graph is already a hard problem, let alone finding one with minimal weight.

Remark

TSP is an **NP-complete** *problem. It is still an open question that whether or not there exists an efficient polynomial time algorithm for its solution. Here "Polynomial time" means functions with complexity* $\mathcal{O}$ *of polynomial functions of the inputs* $|\mathcal{V}|$ *and* $|\mathcal{E}|$.

Shortest Path between A and B

Remark

Fortunately, most of us are not salespersons. In many cases, we simply need to travel from A to B. Then we want to optimize our trip. For example, some students attended a career fair in another state a few days ago. When planning the trip, they may aim at:

- *the cheapest flight;*
- *the flight with least time duration;*
- *the shortest route for driving.*

Shortest Path between A and B

Remark

Fortunately, most of us are not salespersons. In many cases, we simply need to travel from A to B. Then we want to optimize our trip. For example, some students attended a career fair in another state a few days ago. When planning the trip, they may aim at:

- *the cheapest flight;*
- *the flight with least time duration;*
- *the shortest route for driving.*

Remark

We have an abstraction about this type of problems.

Shortest Path Problems

Definition

In a weighted graph, the **shortest path** *between two vertices is a path connecting them of least weight. The* **shortest path problem** *is to find a shortest path between two given distinct vertices, or between all pairs of distinct vertices.*

Shortest Path Problems

Definition

In a weighted graph, the **shortest path** *between two vertices is a path connecting them of least weight. The* **shortest path problem** *is to find a shortest path between two given distinct vertices, or between all pairs of distinct vertices.*

Remark

You may have the stereotype that if two vertices are connected, then the direct edge would be the solution.

Shortest Path Problems

Definition

In a weighted graph, the **shortest path** *between two vertices is a path connecting them of least weight. The* **shortest path problem** *is to find a shortest path between two given distinct vertices, or between all pairs of distinct vertices.*

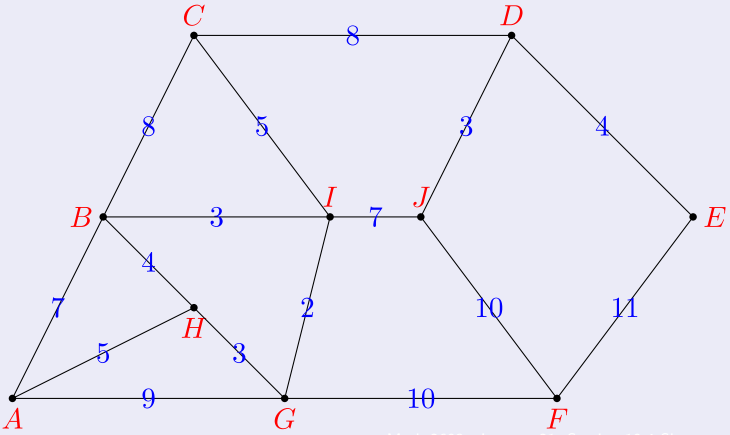
Remark

You may have the stereotype that if two vertices are connected, then the direct edge would be the solution. It's not always true. For example, ticket fares of flights have numerous counterexamples. So we need systematic methods.

An example

Example

What's the shortest path between A and E?



The difficulty

Remark

The main difficulty of this problem is the multiple paths between two vertices. Apparently, some paths are worse than others, but we don't have a clear heuristic to eliminate them.

The difficulty

Remark

The main difficulty of this problem is the multiple paths between two vertices. Apparently, some paths are worse than others, but we don't have a clear heuristic to eliminate them.

Remark

If we apply the "brute force" method instead, in the worst case, say the graph is isomorphic to K_n, the number of paths would be nearly $n!$, which is too big.

The difficulty

Remark

The main difficulty of this problem is the multiple paths between two vertices. Apparently, some paths are worse than others, but we don't have a clear heuristic to eliminate them.

Remark

If we apply the "brute force" method instead, in the worst case, say the graph is isomorphic to K_n*, the number of paths would be nearly* $n!$*, which is too big.*

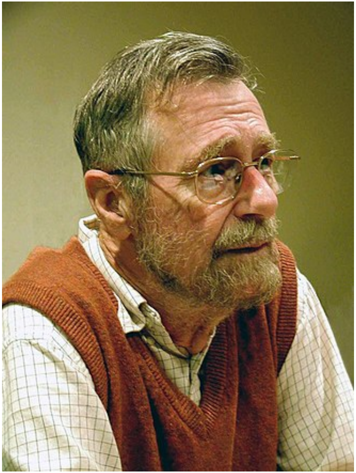
Remark

So we need something smarter.

Dijkstra's Algorithms

Dijkstra's contribution

Edsger Wybe Dijkstra (1930-2002) was a Dutch systems scientist, programmer, software engineer, science essayist, and pioneer in computing science. He came up with the famous algorithm in 1956.
Photo is from Wikipedia.



The algorithm

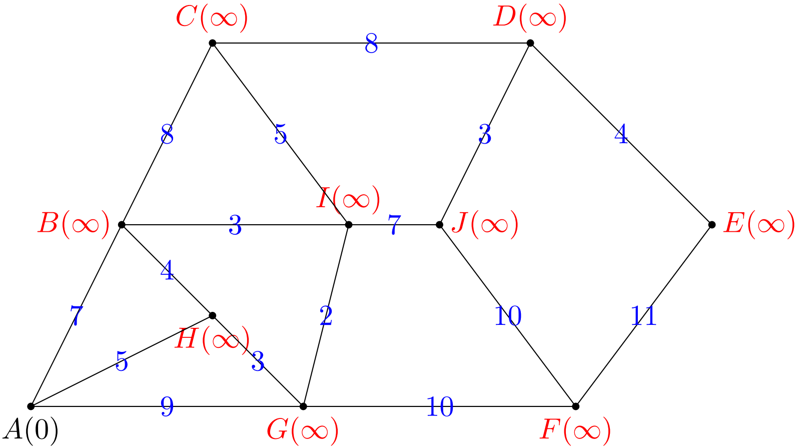
Our input is a weighted graph and we want to find the shortest paths from A to all other vertices. The idea is to add temporary and permanent labels at the vertices. A permanent labels means the distance of the shortest path from A.

The algorithm

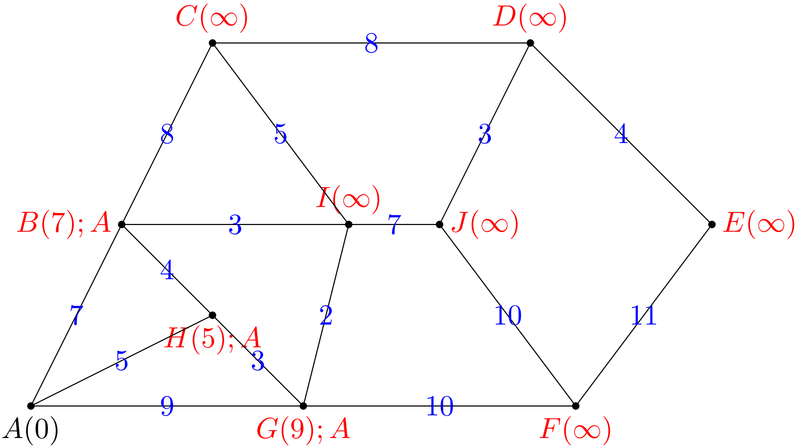
Our input is a weighted graph and we want to find the shortest paths from A to all other vertices. The idea is to add temporary and permanent labels at the vertices. A permanent labels means the distance of the shortest path from A.

1. Add permanent label 0 at A; add temporary labels ∞ at all other vertices.
2. While there exists at least one temporary label: take the vertex v_i with the latest permanent label d (at beginning the vertex is A). For all edges $\{v_i, v\}$ incident to v_i, if v has not got a permanent label, we compare the label at v and $d + w(v_i, v)$. If the latter number is smaller, then we update the temporary label at v to be it. In addition, we set the predecessor of v to be v_i.
3. Among all temporary labels, we pick a smallest, say at v'. We turn this label at v' to be permanent. Repeat (2).

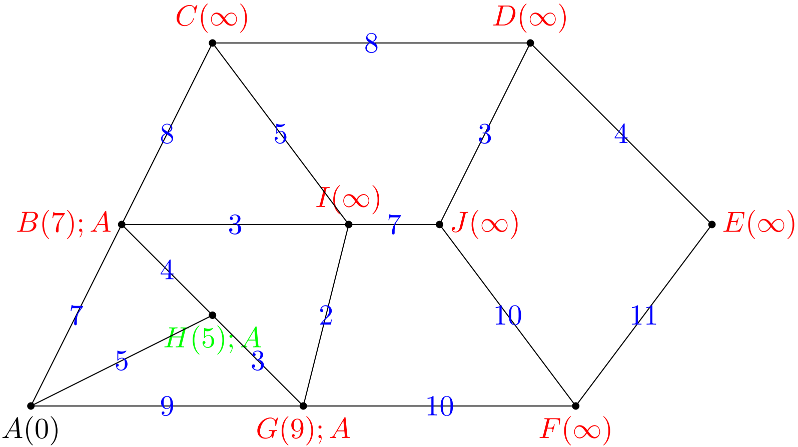
Example: Dijkstra's algorithm



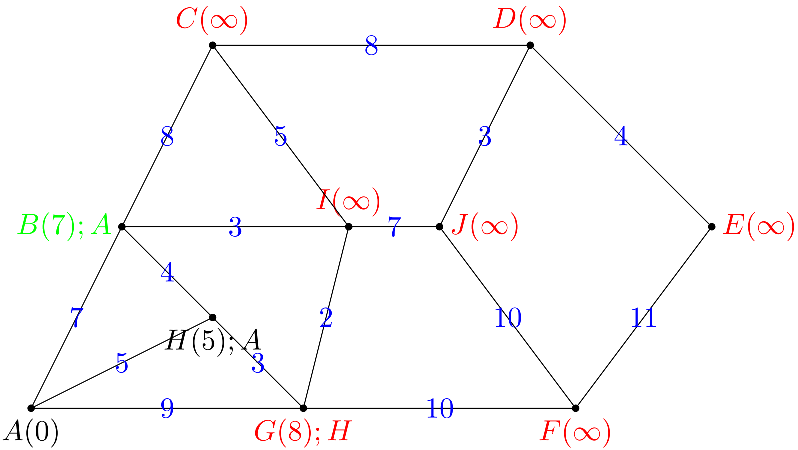
Example: Dijkstra's algorithm



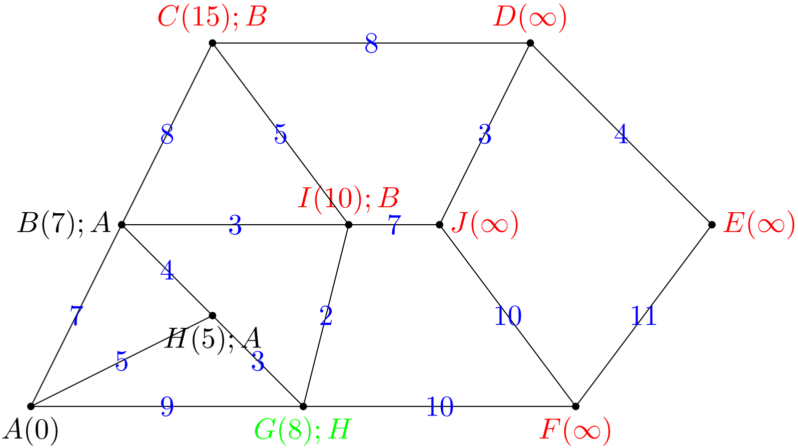
Example: Dijkstra's algorithm



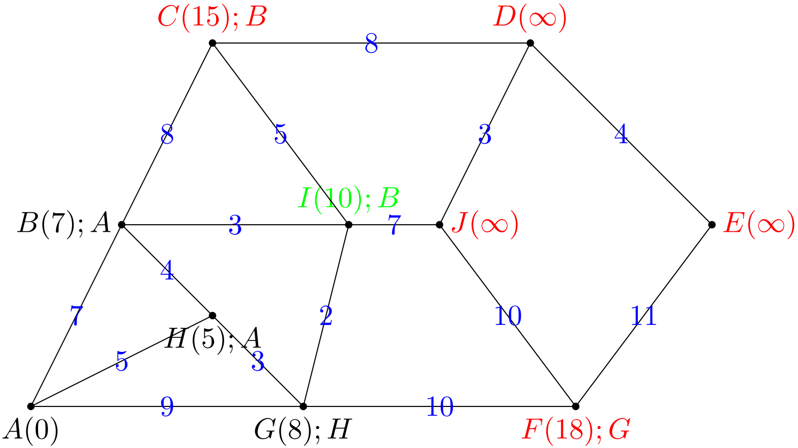
Example: Dijkstra's algorithm



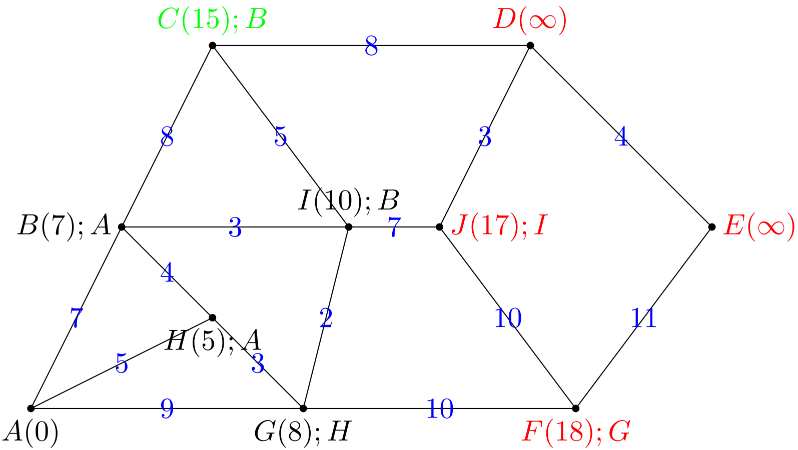
Example: Dijkstra's algorithm



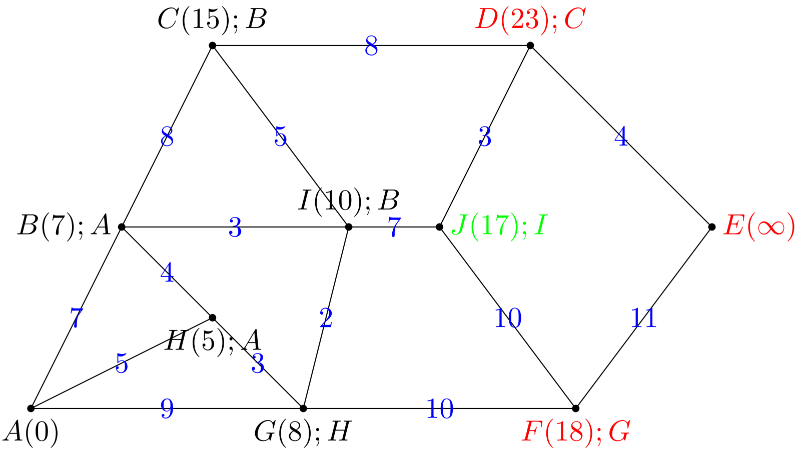
Example: Dijkstra's algorithm



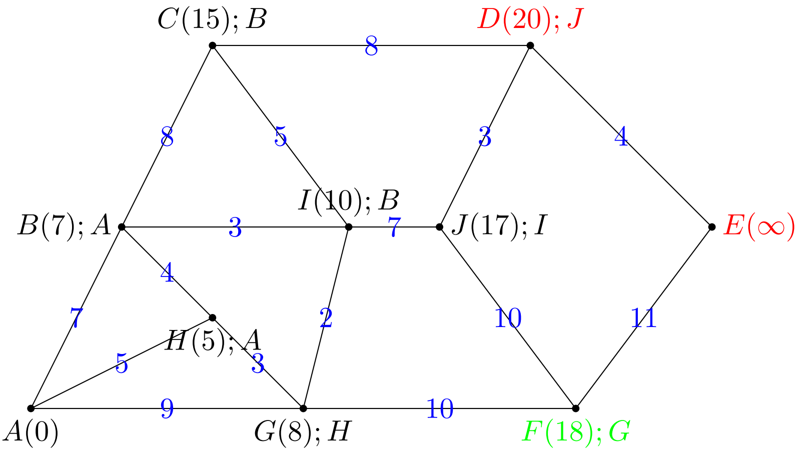
Example: Dijkstra's algorithm



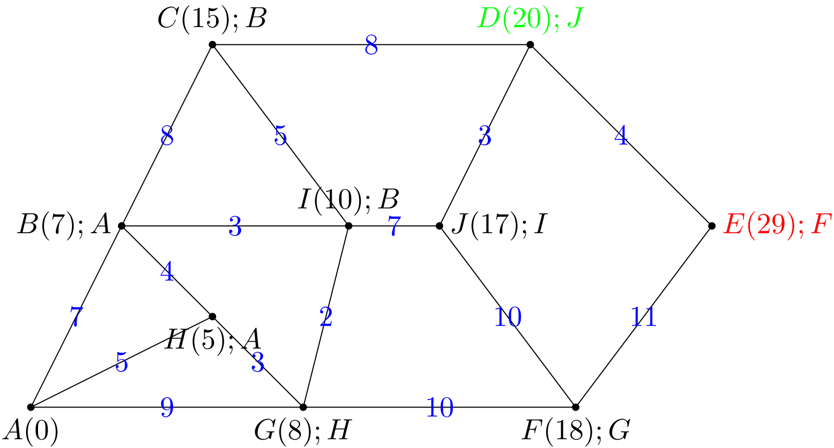
Example: Dijkstra's algorithm



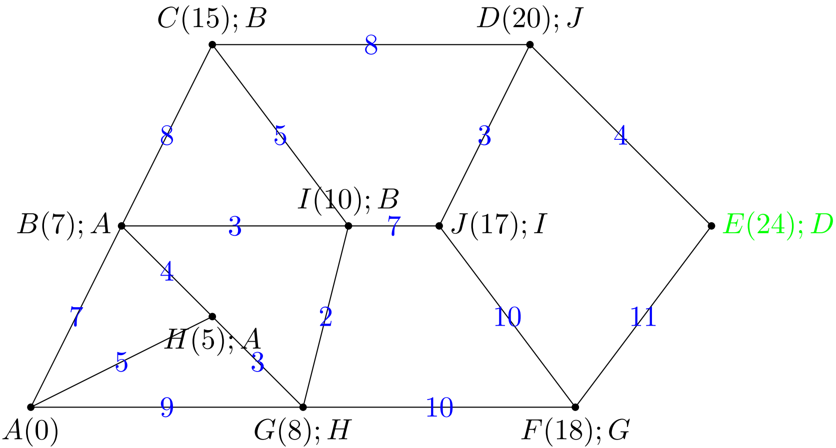
Example: Dijkstra's algorithm



Example: Dijkstra's algorithm



Example: Dijkstra's algorithm



The result

Remark

The shortest path from A to E is

$$ABIJDE.$$

The weight is

$$24 = 7 + 3 + 7 + 3 + 4.$$

The output and complexity

Remark

If we want the shortest path from A to E, we can track the predecessors from E, all the way back to A. The weight is just the permanent label at E.

The output and complexity

Remark

If we want the shortest path from A to E, we can track the predecessors from E, all the way back to A. The weight is just the permanent label at E.

Remark

Suppose the graph has n vertices. Each round we add one more permanent label. So we need up to n rounds.

The output and complexity

Remark

If we want the shortest path from A to E, we can track the predecessors from E, all the way back to A. The weight is just the permanent label at E.

Remark

Suppose the graph has n vertices. Each round we add one more permanent label. So we need up to n rounds. In the i-th round, we do additions $d + w(v_i, v)$ and comparisons up to $3(n - i)$ times, so the complexity within each round is $\mathcal{O}(n)$. In summary, the complexity of Dijkstra's algorithm to find the shortest paths from A to all other vertices is $\mathcal{O}(n^2)$.

The proof of Dijkstra's Algorithm

Proof.

It suffices to prove that each time when we convert a temporary label to a permanent one, the label is indeed the weight of a shortest path. We apply induction on the number of permanent labels. Suppose we already have permanent labels at $A = v_0, v_1, \ldots, v_k$ $(k \geq 0)$ and now v_{k+1} has the smallest temporary label.

The proof of Dijkstra's Algorithm

Proof.

It suffices to prove that each time when we convert a temporary label to a permanent one, the label is indeed the weight of a shortest path. We apply induction on the number of permanent labels. Suppose we already have permanent labels at $A = v_0, v_1, \ldots, v_k$ $(k \geq 0)$ and now v_{k+1} has the smallest temporary label. We pick an arbitrary path from A to v_{k+1}, say

$$p_0 = A - p_1 - \ldots - p_m - p_{m+1} = v_{k+1}.$$

(to be continued) □

The proof of Dijkstra's Algorithm

Proof.

Since A has a permanent label, we can find the largest index j such that p_j has a permanent label. Then $0 \leq j \leq m$ and p_{j+1} has a temporary label.

The proof of Dijkstra's Algorithm

Proof.

Since A has a permanent label, we can find the largest index j such that p_j has a permanent label. Then $0 \leq j \leq m$ and p_{j+1} has a temporary label.Since p_j has a permanent label, by the induction hypothesis, the weight of $p_0 = A - p_1 - \ldots - p_j$ is at least the label at p_j. So when the label at p_j is convert to a permanent one, in the next round, the edge $p_j p_{j+1}$ was considered. Hence the label at p_{j+1} is at most the label at p_j plus the weight of $p_j p_{j+1}$, which equals to the weight of $p_0 = A - p_1 - \ldots - p_{j+1}$!

The proof of Dijkstra's Algorithm

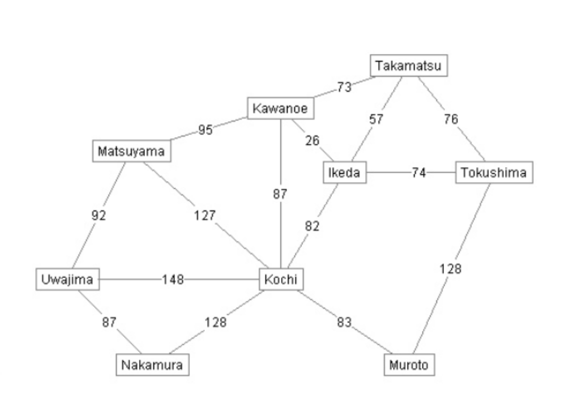
Proof.

Since A has a permanent label, we can find the largest index j such that p_j has a permanent label. Then $0 \leq j \leq m$ and p_{j+1} has a temporary label.Since p_j has a permanent label, by the induction hypothesis, the weight of $p_0 = A - p_1 - \ldots - p_j$ is at least the label at p_j. So when the label at p_j is convert to a permanent one, in the next round, the edge $p_j p_{j+1}$ was considered. Hence the label at p_{j+1} is at most the label at p_j plus the weight of $p_j p_{j+1}$, which equals to the weight of $p_0 = A - p_1 - \ldots - p_{j+1}$!
Finally the weight of the path from A to v_{k+1} is at least the weight of $p_0 = A - p_1 - \ldots - p_{j+1}$, no less than the temporary label at p_{j+1}, which is no less than the temporary label at p_{k+1}, by the choice of p_{k+1}. The inductive step is done. □

Another example

Example

Find the shortest path from Nakamura to Tokushima.



The answer

Solution

The shortest path is

$$Nakamura - Kochi - Ikeda - Tokushima.$$

The weight is $128 + 82 + 74 = 284$.

Homework Assignment #12 - today

Section 10.4 Exercise 1, 6, 9.