

2022-11-25

# Tropical Semirings

Tropical Semirings

A general method for declaratively solving graph problems

Simon Zeng

UWISC

November 25, 2022



## └ Handout

- You may obtain a copy of these notes and code samples on my website
- Code samples are in Haskell and were tested on GHC 9.2.4

## Links

- <https://simonzeng.com/tropical.pdf>
- <https://simonzeng.com/tropical.hs>

- Greeting
- Functional programming talk
  - Not lambda
  - Not purity
  - Not monads
- Solving real problems declaratively
  - Graph problems!
- We develop an algebraic method that elegantly describes a class of graph problems and their solutions
- These notes are available on my website

# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ Dijkstra's Shortest Path Algorithm

- Q: read
- (next, then read)
- Everyone love's Dijkstra's
- Set up priority queue, put things into it, take things out of it, stop iterating based on it
- (next, then read)
- It's a very imperative algorithm
- CS 341 shows a few other graph path algorithms
  - from that you'd think they're all inherently imperative
- But: graph problems are **not** inherently imperative!
- Let's zoom in to the meat

## Tropical Semirings

## └ The Essence of the Path Algorithm

## └ Dijkstra's Shortest Path Algorithm

## Dijkstra's Shortest Path Algorithm

- Q: How do we traditionally get the shortest path between two nodes on a graph?
- A: Dijkstra's algorithm

## Pseudocode

```

1 function Dijkstra(Graph, source):
2   dist[source] ← 0
3   create vertex priority queue Q
4   for each vertex v in Graph.Vertices:
5     if v != source
6       dist[v] ← ∞
7       dist[v], v ← DEQUEUE, ENQUEUE
8   while Q is not empty:
9     u ← Q.dequeue()
10    for each neighbor v of u:
11      alt ← dist[u] + length(u, v)
12      if alt < dist[v]:
13        dist[v] ← alt
14        Q.dequeue(v)
15  return dist, prev

```

- Q: read
- (next, then read)
- Everyone love's Dijkstra's
- Set up priority queue, put things into it, take things out of it, stop iterating based on it
- (next, then read)
- It's a very imperative algorithm
- CS 341 shows a few other graph path algorithms
  - from that you'd think they're all inherently imperative
- But: graph problems are **not** inherently imperative!
- Let's zoom in to the meat

## Tropical Semirings

## └ The Essence of the Path Algorithm

## └ Dijkstra's Shortest Path Algorithm

## Dijkstra's Shortest Path Algorithm

- Q: How do we traditionally get the shortest path between two nodes on a graph?
- A: Dijkstra's algorithm

## Pseudocode

```

1 function Dijkstra(Graph, source):
2   dist[source] ← 0
3   create vertex priority queue Q
4   for each vertex v in Graph.Vertices:
5     if v != source
6       dist[v] ← ∞
7   Q.add(source, dist[source])
8   while Q is not empty:
9     u ← Q.extract_min()
10    for each neighbor v of u:
11      alt ← dist[u] + length(u, v)
12      if alt < dist[v]:
13        dist[v] ← alt
14        Q.decrease_priority(v, alt)
15  return dist, prev

```

- Very classic, but:
  - Uses lots of state and mutation
  - Hard to tell what's going on from just reading the code

- Q: read
- (next, then read)
- Everyone love's Dijkstra's
- Set up priority queue, put things into it, take things out of it, stop iterating based on it
- (next, then read)
- It's a very imperative algorithm
- CS 341 shows a few other graph path algorithms
  - from that you'd think they're all inherently imperative
- But: graph problems are **not** inherently imperative!
- Let's zoom in to the meat

2022-11-25

# Tropical Semirings

└ The Essence of the Path Algorithm

└ Core of Dijkstra's Algorithm

Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :

# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :
  - Get the best distance from source to  $u$ ...

# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :
  - Get the best distance from source to  $u$ ...
  - ... add the weight of the edge  $uv$ ...



# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ Core of Dijkstra's Algorithm

#### Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :
  - Get the best distance from source to  $u$ ...
  - ... add the weight of the edge  $uv$ ...
  - ... compare against existing best distance of  $v$ ...

# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ Core of Dijkstra's Algorithm

#### Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :
  - Get the best distance from source to  $u$ ...
  - ... add the weight of the edge  $uv$ ...
  - ... compare against existing best distance of  $v$ ...
  - ... store the **minimum** between our number and what  $v$  already has

# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ Core of Dijkstra's Algorithm

#### Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :
  - Get the best distance from source to  $u$ ...
  - ... **add** the weight of the edge  $uv$ ...
  - ... compare against existing best distance of  $v$ ...
  - ... store the **minimum** between our number and what  $v$  already has

#### Pseudocode

```
14 add ← Shortest ← Graph.Algorithm, 0  
15 for all  $v$  in  $V$  do  
16   dist[v] ← ∞
```

## Tropical Semirings

## └ The Essence of the Path Algorithm

## └ Core of Dijkstra's Algorithm

## Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :
  - Get the best distance from source to  $u$ ...
  - ... **add** the weight of the edge  $uv$ ...
  - ... compare against existing best distance of  $v$ ...
  - ... store the **minimum** between our number and what  $v$  already has

## Pseudocode

```

14 dist ← dist[u] + Graph.Edge(u, v)
15 if dist < dist[v]
16   dist[v] ← dist

```

- The rest of Dijkstra's tells us **only when** we look at a particular node

## Tropical Semirings

## └ The Essence of the Path Algorithm

## └ Core of Dijkstra's Algorithm

## Core of Dijkstra's Algorithm

- When going from some node  $u$  to some node  $v$ :
  - Get the best distance from source to  $u$ ...
  - ... **add** the weight of the edge  $uv$ ...
  - ... compare against existing best distance of  $v$ ...
  - ... store the **minimum** between our number and what  $v$  already has

## Pseudocode

```

14 dist ← dist[u] + Graph.Edge(u, v)
15 if dist < dist[v]
16   dist[v] ← dist

```

- The rest of Dijkstra's tells us **only when** we look at a particular node
- Dijkstra's is just node ordering boilerplate around this core operation

2022-11-25

# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ A Functional Kernel

A Functional Kernel

#### Original Pseudocode

```
14  $dist \leftarrow dist[0]$  //  $dist[0]$  is length of path to  $u$   
15  $dist[u] \leftarrow dist[0]$   
16  $dist[0] \leftarrow \infty$ 
```

## Tropical Semirings

## └ The Essence of the Path Algorithm

## └ A Functional Kernel

## Original Pseudocode

```
14 dist ← dist[0] + Graph.EdgeCost, 0  
15 for dist ← dist[0]  
16   dist[0] ← dist
```

- The original calculates, then compares, then (sometimes) sets

# Tropical Semirings

## └ The Essence of the Path Algorithm

### └ A Functional Kernel

**Original Pseudocode**

```
14  $dist \leftarrow \infty$ ;  $dist[0] \leftarrow \text{length}(edges[0])$   
15  $dist[0] \leftarrow \min(dist[0], \dots)$ 
```

- The original calculates, then compares, then (sometimes) sets
- The comparison+set can be written as a single function call



## Tropical Semirings

## └ The Essence of the Path Algorithm

## └ A Functional Kernel

## A Functional Kernel

## Original Pseudocode

```

14  $dist \leftarrow dist[2] + \text{Graph.Edge}(u, v)$ 
15  $dist \leftarrow \min(dist, dist[2])$ 
16  $dist[2] \leftarrow dist$ 

```

- The original calculates, then compares, then (sometimes) sets
- The comparison+set can be written as a single function call

## Refined pseudocode

```

dist[2] ← min(dist[2], dist[2] + Graph.Edge(u, v))

```

## Tropical Semirings

## └ The Essence of the Path Algorithm

## └ A Functional Kernel

## A Functional Kernel

## Original Pseudocode

```

14  $dist \leftarrow dist[2] + \text{Graph.Edge}(u, v)$ 
15  $dist \leftarrow \min(dist)$ 
16  $dist[2] \leftarrow dist$ 

```

- The original calculates, then compares, then (sometimes) sets
- The comparison+set can be written as a single function call

## Refined pseudocode

```

 $dist[2] \leftarrow \min(dist[2], dist[2] + \text{Graph.Edge}(u, v))$ 

```

- This is the core of path algorithms!

2022-11-25

# Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

The Algebra of the Path Algorithm

Essence of the path algorithm

$dist[v] \leftarrow \min(dist[v], dist[u] + \text{Graph.Edge}(u, v))$

2022-11-25

# Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

The Algebra of the Path Algorithm

Essence of the path algorithm

$dist[v] \leftarrow \min(dist[v], dist[u] + Graph.Algebraic[v])$

- There are only two operations here:  $\min$  and  $+$

2022-11-25

# Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

### The Algebra of the Path Algorithm

#### Essence of the path algorithm

$dist[v] \leftarrow \min_{(u,v) \in E} \{ dist[u] + \text{Graph.Edge}(u,v) \}$

- There are only two operations here:  $\min$  and  $+$
- Q: What algebraic structures have two operations?

# Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

## Essence of the path algorithm

 $short[u] \leftarrow \min(short[u], short[v] + \text{Graph.Edge}(u, v))$ 

- There are only two operations here:  $\min$  and  $+$
- Q: What algebraic structures have two operations?
- A: Rings, fields, and related structures

# Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

## Essence of the path algorithm

```
shortestPath(s, t) = shortest(s, t, shortestPath, +)
```

- There are only two operations here:  $\min$  and  $+$
- Q: What algebraic structures have two operations?
- A: Rings, fields, and related structures
- Let's define a ring-like structure  $(R, +, \cdot)$  such that:

## Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

## Essence of the path algorithm

```
shortestPath(s, t) = shortest(s, t, shortestPath, +, *)
```

- There are only two operations here:  $\min$  and  $+$
- Q: What algebraic structures have two operations?
- A: Rings, fields, and related structures
- Let's define a ring-like structure  $(R, +, \cdot)$  such that:
  - The underlying set  $R$  is  $\mathbb{R}$



## Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

## Essence of the path algorithm

```
dist[u] ← min(dist[u], dist[v] + Graph.Edges[v, u])
```

- There are only two operations here:  $\min$  and  $+$
- Q: What algebraic structures have two operations?
- A: Rings, fields, and related structures
- Let's define a ring-like structure  $(R, +, \cdot)$  such that:
  - The underlying set  $R$  is  $\mathbb{R}$
  - The "addition" operation  $+$  is the function  $\min$  that takes the minimum of its two arguments

## Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

## Essence of the path algorithm

```
dist[u] ← min(dist[u], dist[v] + Graph.Edge(u, v))
```

- There are only two operations here:  $\min$  and  $+$
- Q: What algebraic structures have two operations?
- A: Rings, fields, and related structures
- Let's define a ring-like structure  $(R, +, \cdot)$  such that:
  - The underlying set  $R$  is  $\mathbb{R}$
  - The "addition" operation  $+$  is the function  $\min$  that takes the minimum of its two arguments
  - The "multiplication" operation  $\cdot$  is the usual addition on reals

## Tropical Semirings

## └ The Algebra of the Path Algorithm

## └ The Algebra of the Path Algorithm

## The Algebra of the Path Algorithm

## Essence of the path algorithm

$$\text{dist}[v] \leftarrow \min_{u \in \text{Graph.Edges}(v, \cdot)} \{\text{dist}[u] + \text{Graph.Edges}(u, v)\}$$

- There are only two operations here:  $\min$  and  $+$
- Q: What algebraic structures have two operations?
- A: Rings, fields, and related structures
- Let's define a ring-like structure  $(R, +, \cdot)$  such that:
  - The underlying set  $R$  is  $\mathbb{R}$ .
  - The "addition" operation  $+$  is the function  $\min$  that takes the minimum of its two arguments
  - The "multiplication" operation  $\cdot$  is the usual addition on reals

## Essence of the path algorithm, algebraically

$$\text{dist}[v] = \min_{u \in \text{Graph.Edges}(u, v)} \{\text{dist}[u] + \text{Graph.Edges}(u, v)\}$$

# Tropical Semirings

- └ The Algebra of the Path Algorithm

- └ What Structure do we Have?