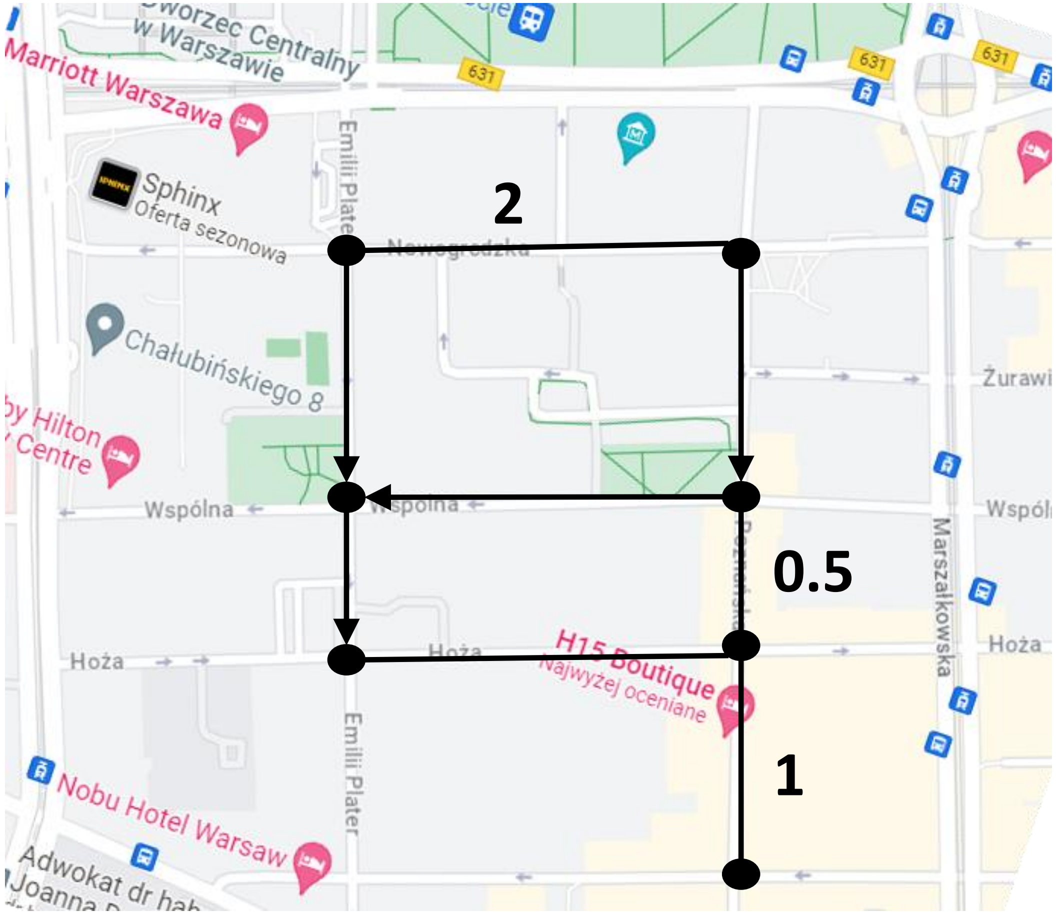
Shortest paths, edge weights, and models of computation

[Adam Karczmarz]

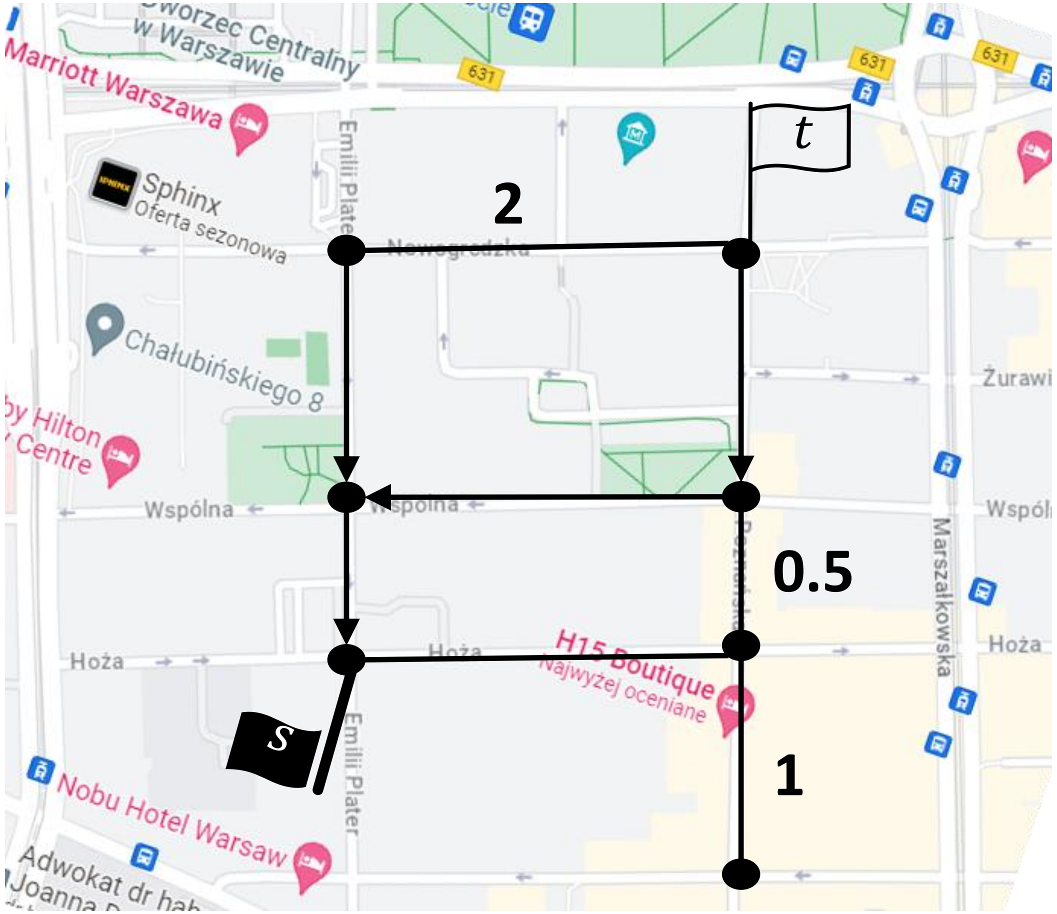
22/01/2026

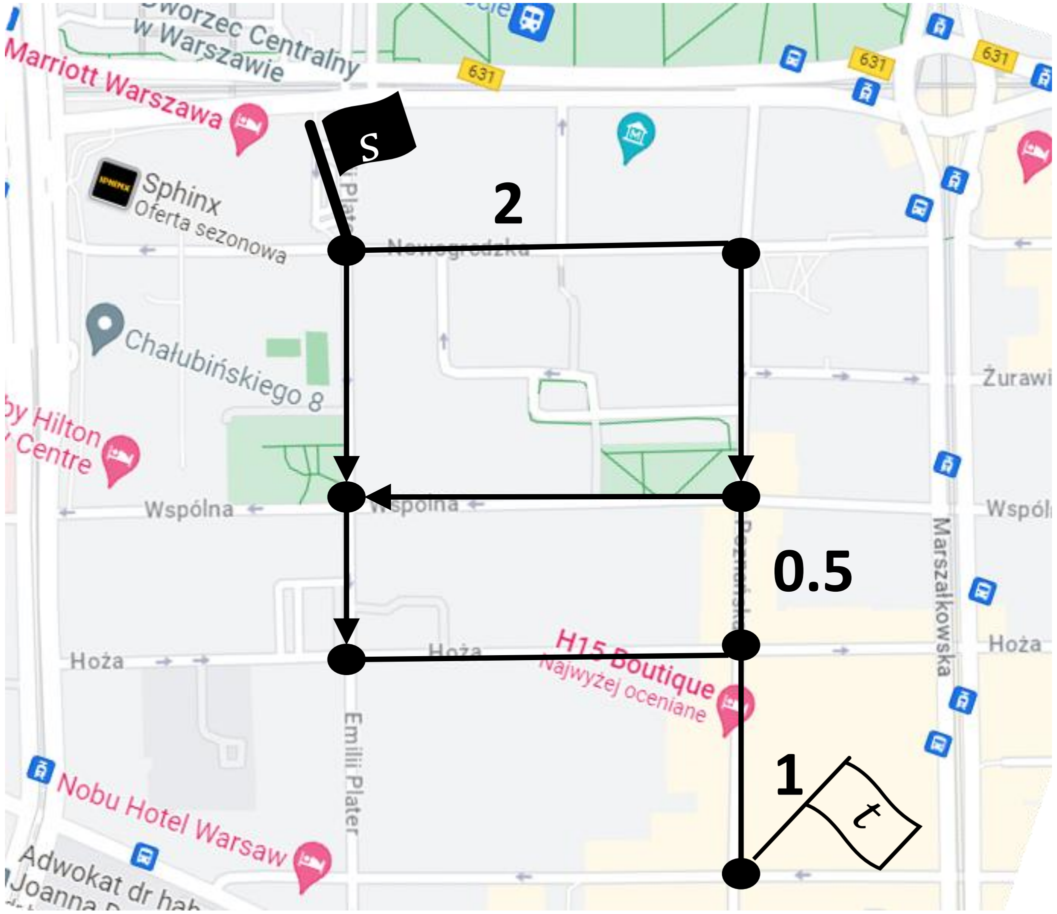
Plan

1. Problem definition.
2. Recap of some basic techniques.
3. Recent breakthroughs.
4. On the way:
    - models of computations,
    - how restricted edge weight domain is exploited,
5. Briefly about one related result of ours.



2
0.5
1
Dworzec Centralny w Warszawie
Marriott Warszawa
631
Sphinx
Oferta sezonowa
Emilii Plater
Nowogrodzka
Chałubińskiego 8
Żurawia
Wspólna
Hoża
Marszałkowska
H15 Boutique
Najwyżej oceniane
Nobu Hotel Warsaw


t
2
0.5
1
s
Marriott Warszawa
Sphinx
Oferta sezonowa
Chałubińskiego 8
Emilii Plater
Nowogrodzka
Wspólna
Hoża
H15 Boutique
Najwyżej oceniane
Nobu Hotel Warsaw
Marszałkowska
Żurawi
631


s
2
0.5
1
t
Dworzec Centralny w Warszawie
Marriott Warszawa
631
Sphinx
Oferta sezonowa
Nowogrodzka
Chałubińskiego 8
by Hilton
Centre
Wspólna
Hoża
H15 Boutique
Najwyżej oceniane
Żurawi
Marszałkowska
Emilii Plater
Nobu Hotel Warsaw
Adwokat dr hab.

The shortest path problem

Input:
- $G = (V, E)$: a weighted directed graph with n vertices and $m \geq n$ edges.
- Two vertices: source/target $s, t \in V$.

Goal:
Compute a shortest $s \rightarrow t$ path in G.

The shortest path problem

Input:

- $G = (V, E)$: a weighted directed graph with n vertices and $m \geq n$ edges.
- Two vertices: source/target $s, t \in V$.

Goal:

Compute a shortest $s \to t$ path in G.

We want to solve the problem:

- exactly,

The shortest path problem

Input:

- $G = (V, E)$: a weighted directed graph with n vertices and $m \geq n$ edges.
- Two vertices: source/target $s, t \in V$.

Goal:

Compute a shortest $s \rightarrow t$ path in G.

We want to solve the problem:

- exactly,
- fast in the worst case (asymptotically).

SSSP

Some facts:

- In general, solving the problem for all targets $t \in V$ does not look any easier.

SSSP

Some facts:

- In general, solving the problem for all targets $t \in V$ does not look any easier.
- Finding paths not easier than computing optimal path lengths (distances).

SSSP

Some facts:

- In general, solving the problem for all targets $t \in V$ does not look any easier.
- Finding paths not easier than computing optimal path lengths (distances).

Single-Source Shortest Paths (SSSP): [vague]
Given a weighted directed graph $G = (V, E)$ with n vertices and m edges, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.

SSSP

Some facts:

- In general, solving the problem for all targets $t \in V$ does not look any easier.
- Finding paths not easier than computing optimal path lengths (distances).

Single-Source Shortest Paths (SSSP): [vague]
Given a weighted directed graph $G = (V, E)$ with n vertices and m edges, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.

Well-defined:
$|\mathrm{dist}_G(u, v)| \neq \infty$ for all $u, v \in V$
Equiv: every pair reachable
no negative-weight cycles in G.

SSSP

Some facts:

- In general, solving the problem for all targets $t \in V$ does not look any easier.
- Finding paths not easier than computing optimal path lengths (distances).

Single-Source Shortest Paths (SSSP): [vague]
Given a weighted directed graph $G = (V, E)$ with n vertices and m edges, and a source $s \in V$, compute $\text{dist}_G(s, t)$ for all $t \in V$.

Notation:

- $\text{poly}(n)$, $\text{poly}(n, m)$; $\text{polylog}(n) = \log^{O(1)} n$,
- $\tilde{O}\big(f(n, m)\big) := O\big(f(n, m)\text{polylog}(n)\big)$.
- $w(uv) :=$ weight of a (directed) edge uv.

Well-defined:
$|\text{dist}_G(u, v)| \neq \infty$ for all $u, v \in V$
Equiv: every pair reachable
no negative-weight cycles in G.

(obsolete) Manual for solving SSSP

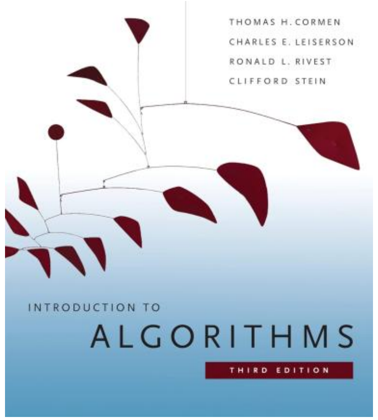
Single-Source Shortest Paths (SSSP): [vague]
Given a weighted directed graph $G = (V, E)$ with n vertices and m edges, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.

Does G have only non-negative weights?

(obsolete) Manual for solving SSSP

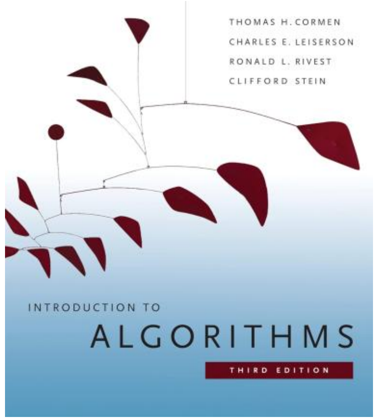
<u>Single-Source Shortest Paths (SSSP):</u> [vague]
Given a weighted directed graph $G = (V, E)$ with n vertices and m edges, and a source $s \in V$, compute $\text{dist}_G(s, t)$ for all $t \in V$.

Does G have only <u>non-negative</u> weights?



(obsolete) Manual for solving SSSP

Single-Source Shortest Paths (SSSP): [vague]
Given a weighted directed graph $G = (V, E)$ with n vertices and m edges, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.



Does G have only non-negative weights?

Yes? Use Dijkstra's algorithm ('56).
$O(m + n \log n)$ time [FT'86].

Or $O(m \log n)$ time without fancy data structures.

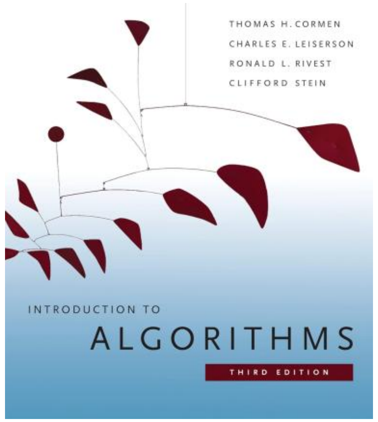
(obsolete) Manual for solving SSSP

<u>Single-Source Shortest Paths (SSSP):</u> [vague]
Given a weighted directed graph $G = (V, E)$ with n vertices and m edges, and a source $s \in V$, compute $\text{dist}_G(s, t)$ for all $t \in V$.

Does G have only <u>non-negative</u> weights?

Yes? Use Dijkstra's algorithm ('56).
$O(m + n \log n)$ time [FT'86].

No? Use Bellman-Ford algorithm ('56).
$O(m \cdot n)$ time.



Or $O(m \log n)$ time without fancy data structures.

Bellman-Ford recap

- If SSSP is well-defined, then an optimal path $P_{st} = s \to t$ has $< n$ edges.

Bellman-Ford recap

- If SSSP is well-defined, then an optimal path $P_{st} = s \to t$ has $< n$ edges.
- Let P_{sv}^{k} be the shortest out of $s \to v$ paths with at most k edges, i.e., $P_{st} = P_{s,t}^{n}$.

Bellman-Ford recap

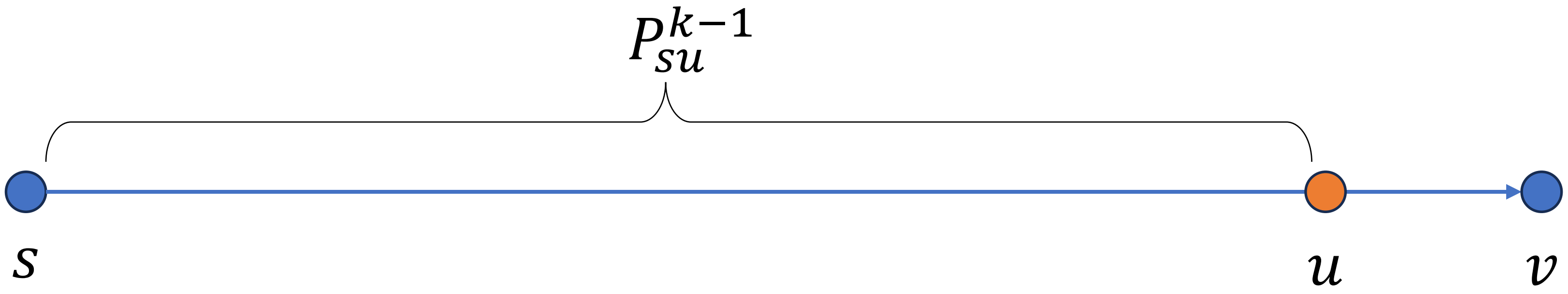
- If SSSP is well-defined, then an optimal path $P_{st} = s \to t$ has $< n$ edges.
- Let P_{sv}^{k} be the shortest out of $s \to v$ paths with at most k edges, i.e., $P_{st} = P_{s,t}^{n}$.
- The (weight of) P_{sv}^{k} can be computed inductively:
  - Either it is an empty path, which can only exist if $s = v$.

Bellman-Ford recap

- If SSSP is well-defined, then an optimal path $P_{st} = s \to t$ has $< n$ edges.
- Let P_{sv}^{k} be the shortest out of $s \to v$ paths with at most k edges, i.e., $P_{st} = P_{s,t}^{n}$.
- The (weight of) P_{sv}^{k} can be computed inductively:
  - ➢ Either it is an empty path, which can only exist if $s = v$.
  - ➢ Or it ends with some edge $uv \in E$, and starts with P_{su}^{k-1}; try all of them.

Bellman-Ford recap

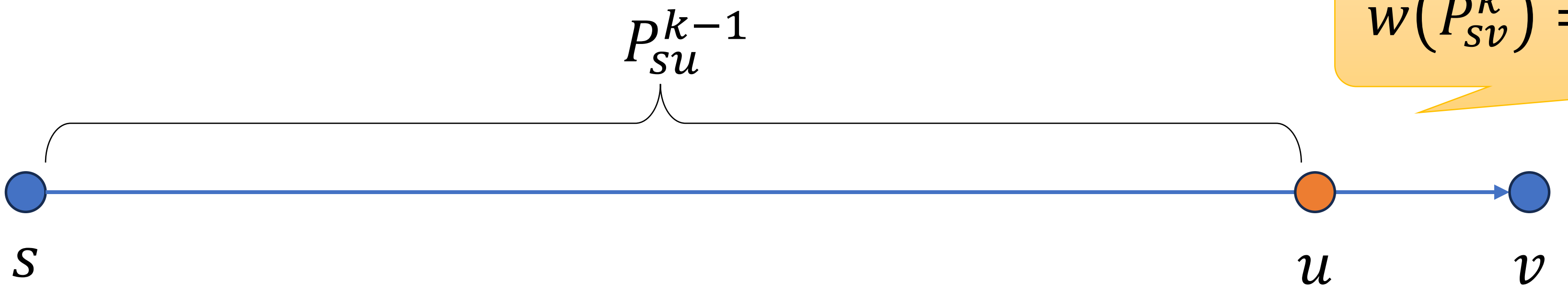
- If SSSP is well-defined, then an optimal path $P_{st} = s \to t$ has $< n$ edges.
- Let P_{sv}^{k} be the shortest out of $s \to v$ paths with at most k edges, i.e., $P_{st} = P_{s,t}^{n}$.
- The (weight of) P_{sv}^{k} can be computed inductively:
  - Either it is an empty path, which can only exist if $s = v$.
  - Or it ends with some edge $uv \in E$, and starts with P_{su}^{k-1}; try all of them.



Bellman-Ford recap

- If SSSP is well-defined, then an optimal path $P_{st} = s \to t$ has $< n$ edges.
- Let P_{sv}^k be the shortest out of $s \to v$ paths with at most k edges, i.e., $P_{st} = P_{s,t}^n$.
- The (weight of) P_{sv}^k can be computed inductively:
  - Either it is an empty path, which can only exist if $s = v$.
  - Or it ends with some edge $uv \in E$, and starts with P_{su}^{k-1}; try all of them.

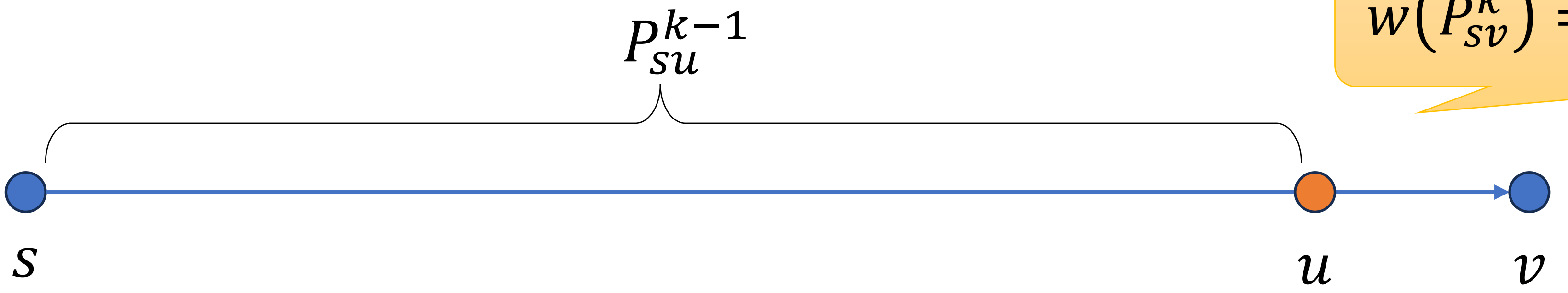
$$w\left(P_{sv}^k\right) = \min_{uv \in E}\left(w\left(P_{su}^{k-1}\right) + w(uv)\right)$$



Bellman-Ford recap

- If SSSP is well-defined, then an optimal path $P_{st} = s \to t$ has $< n$ edges.
- Let P_{sv}^{k} be the shortest out of $s \to v$ paths with at most k edges, i.e., $P_{st} = P_{s,t}^{n}$.
- The (weight of) P_{sv}^{k} can be computed inductively:
  - Either it is an empty path, which can only exist if $s = v$.
  - Or it ends with some edge $uv \in E$, and starts with P_{su}^{k-1}; try all of them.

$$w\left(P_{sv}^{k}\right) = \min_{uv \in E}\left(w\left(P_{su}^{k-1}\right) + w(uv)\right)$$



All weights of P_{st}^{k} for $t \in V$, $k = 0, \dots, n$ and can be computed in $O(m \cdot n)$ time.

Dijkstra recap (non-negative weights)

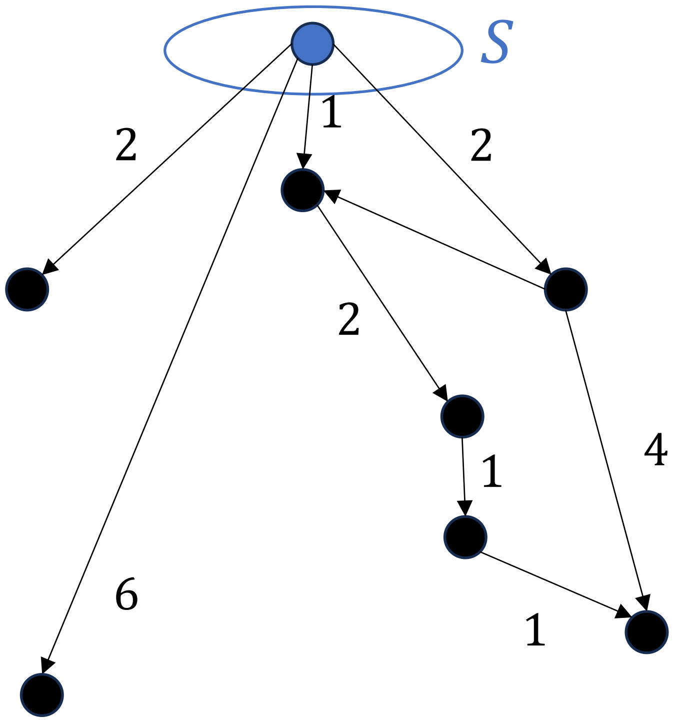
- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.

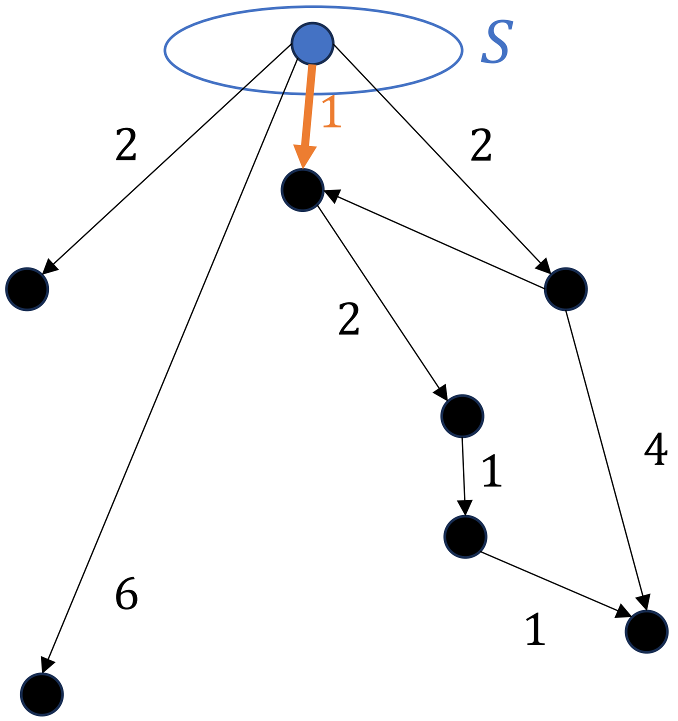
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



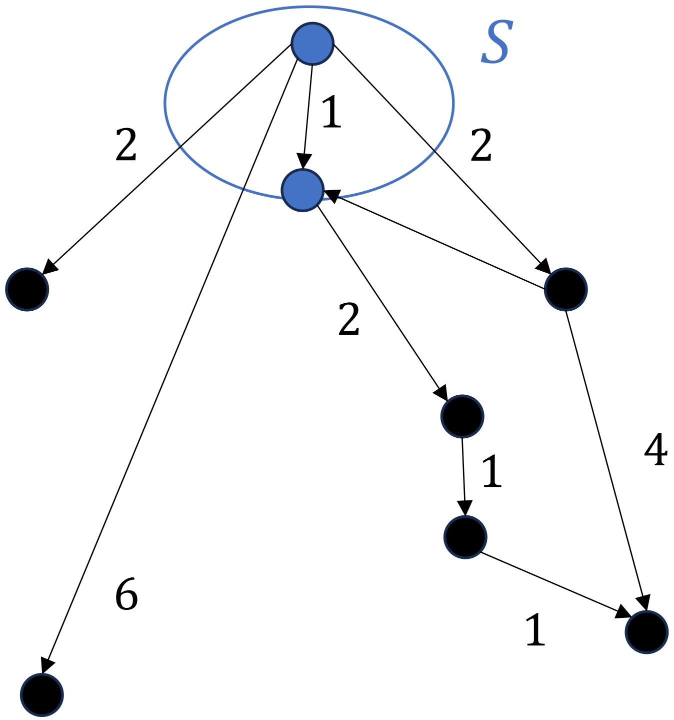
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.



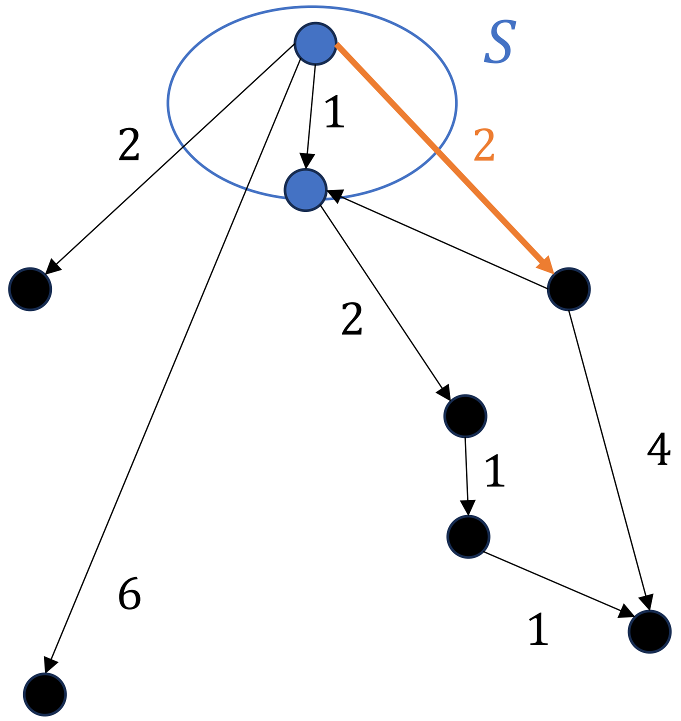
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k + 1)$-th nearest vertex is the one „closest via a single edge" from S.



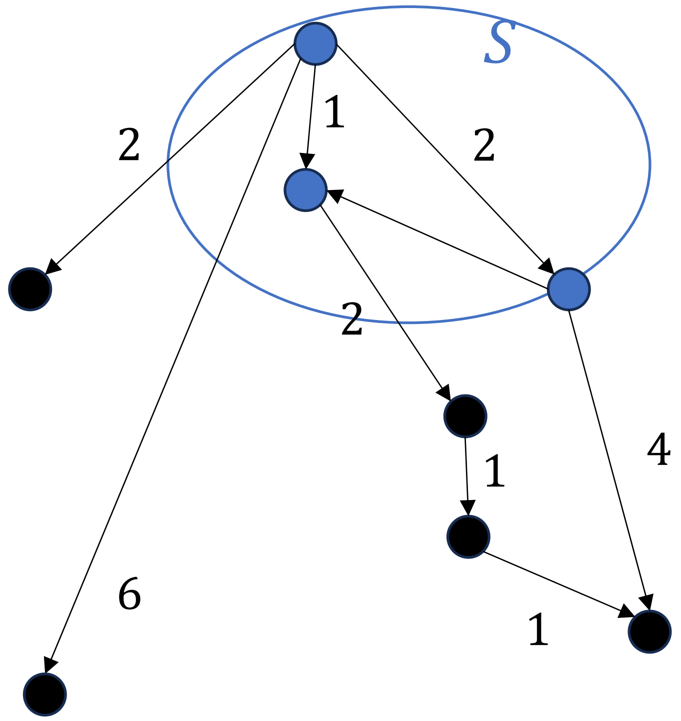
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k + 1)$-th nearest vertex is the one „closest via a single edge” from S.



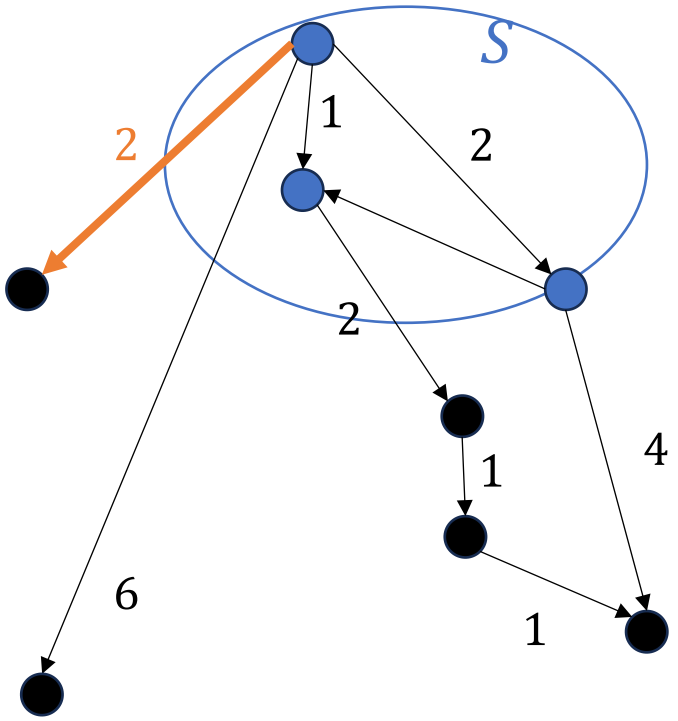
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k + 1)$-th nearest vertex is the one „closest via a single edge” from S.



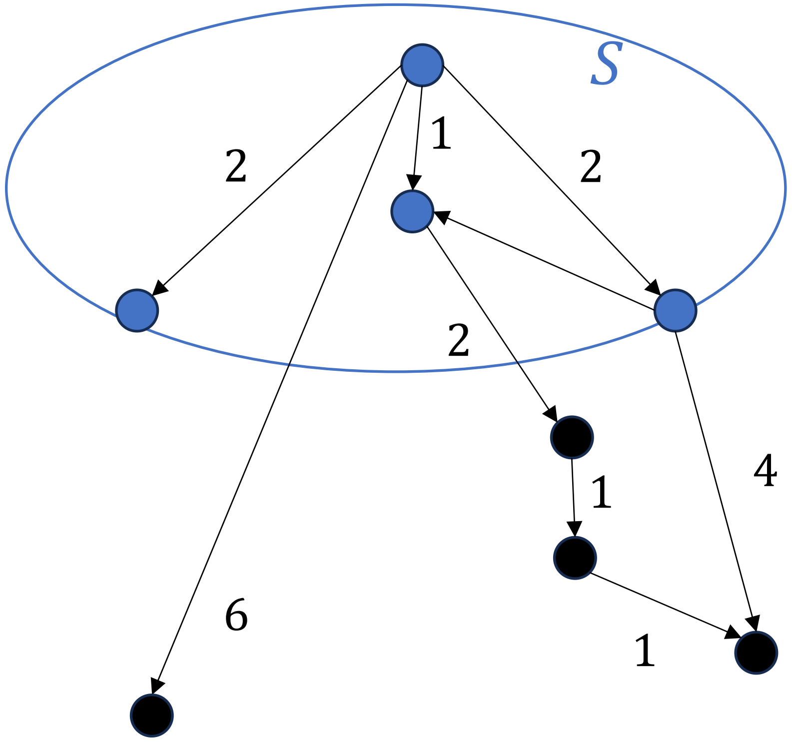
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



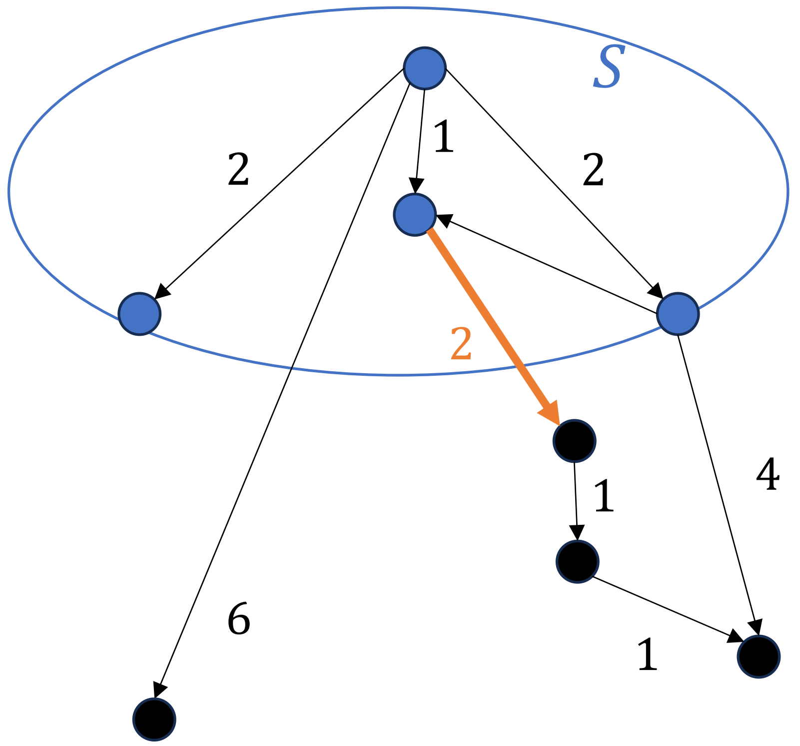
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.



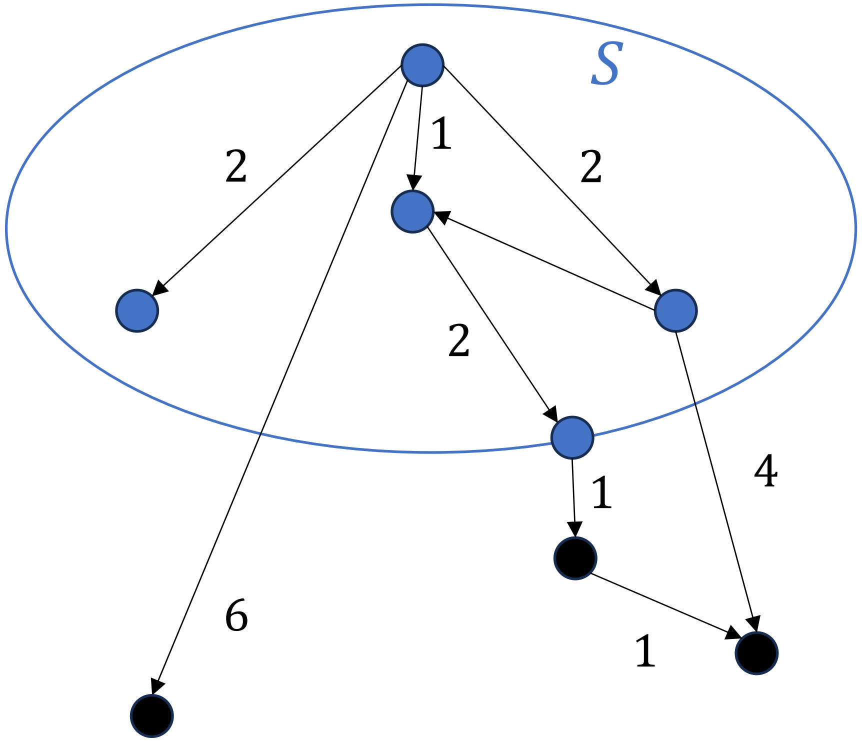
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



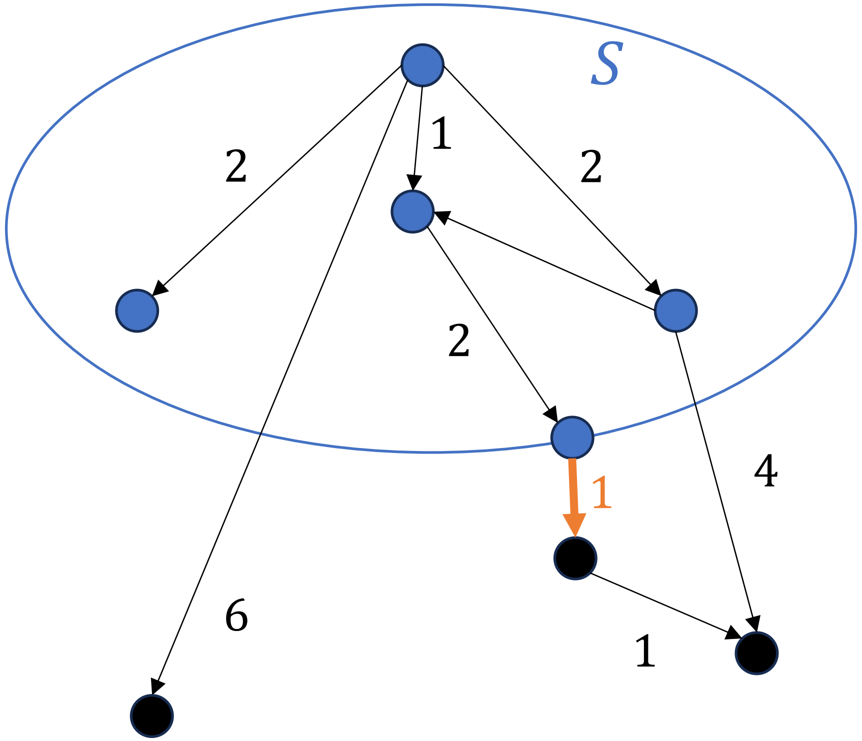
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



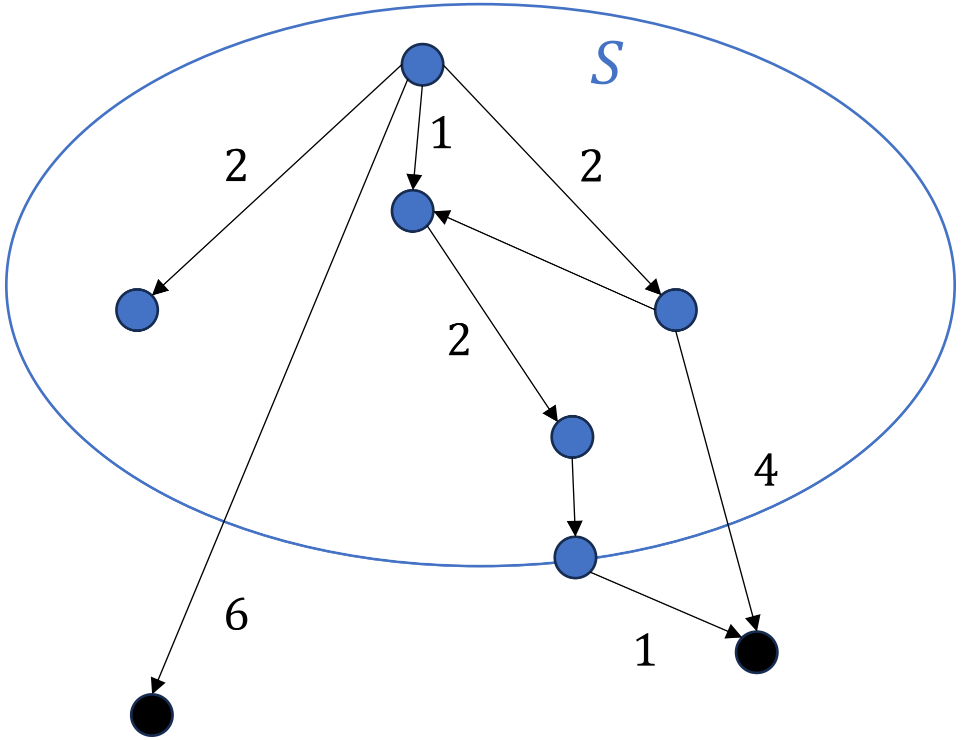
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



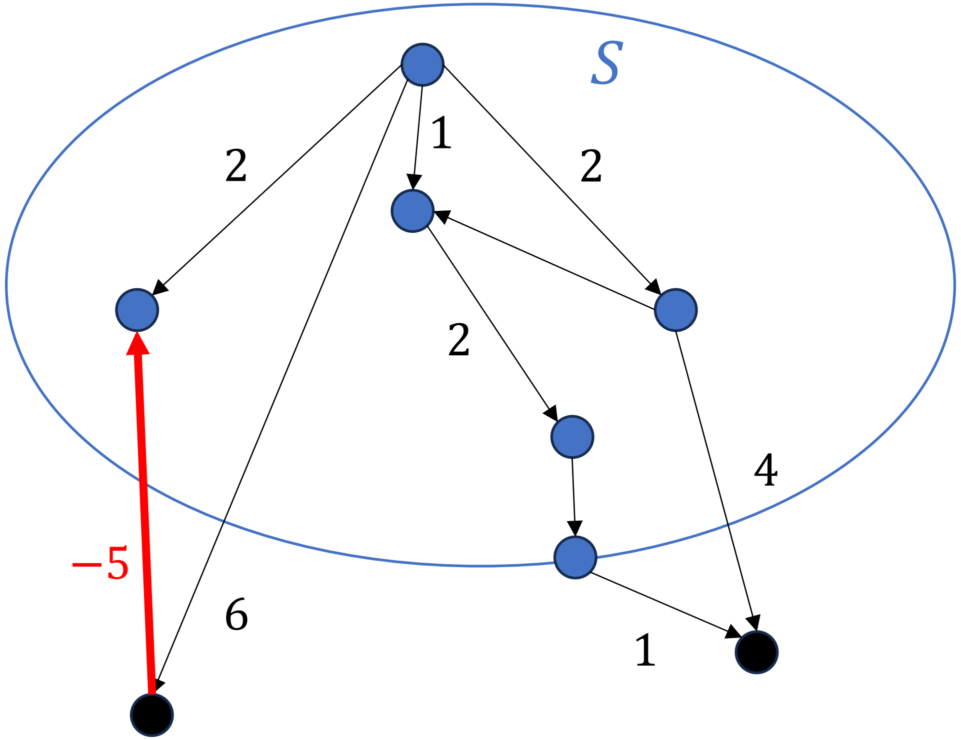
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.



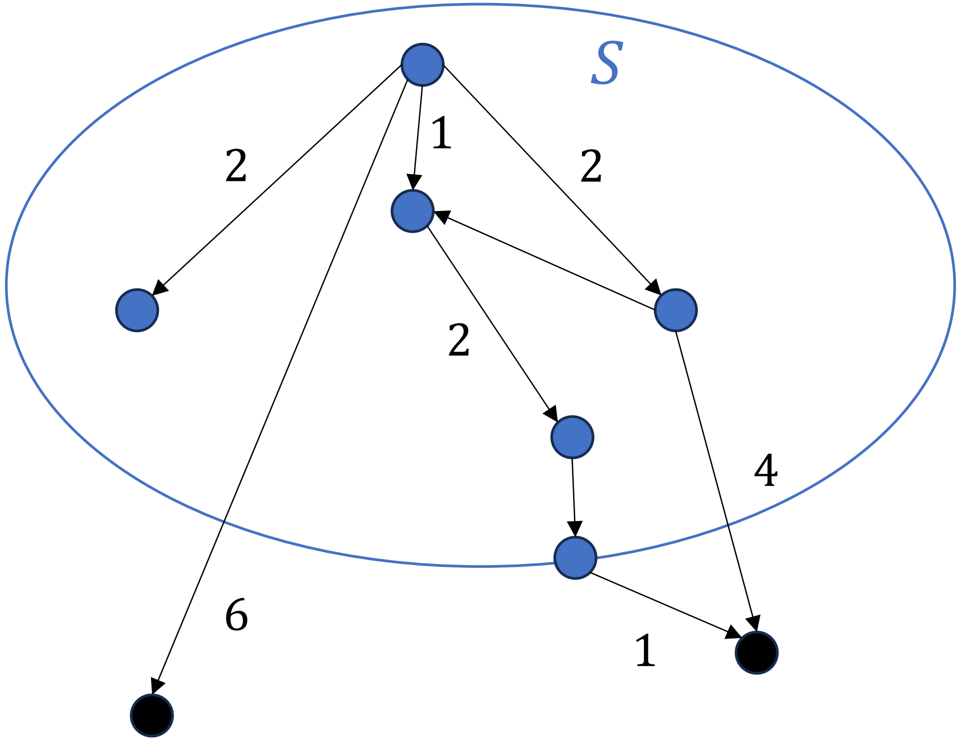
Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.



Dijkstra recap (non-negative weights)

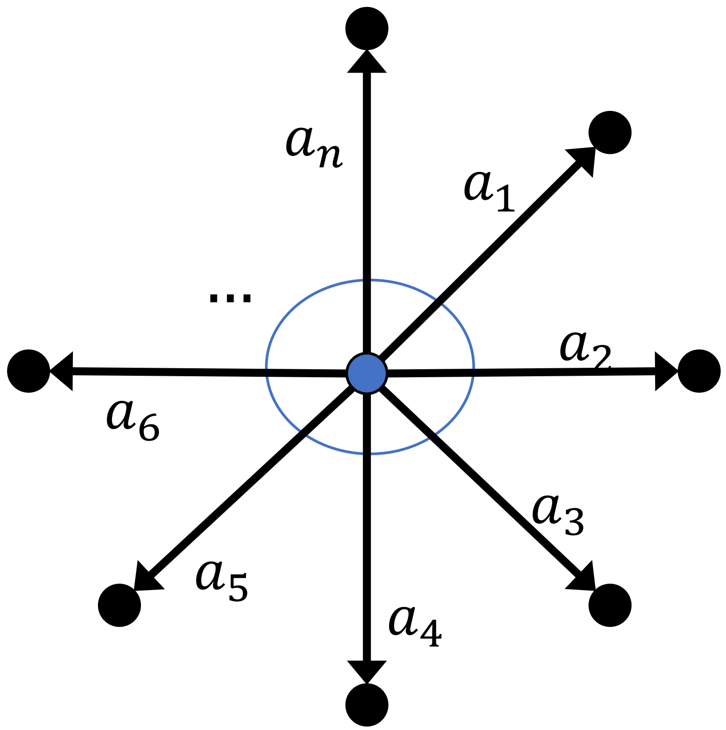
- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



A data structure called a <u>priority queue</u> allows finding the next nearest vertex in $O(\log n)$ time.

Dijkstra recap (non-negative weights)

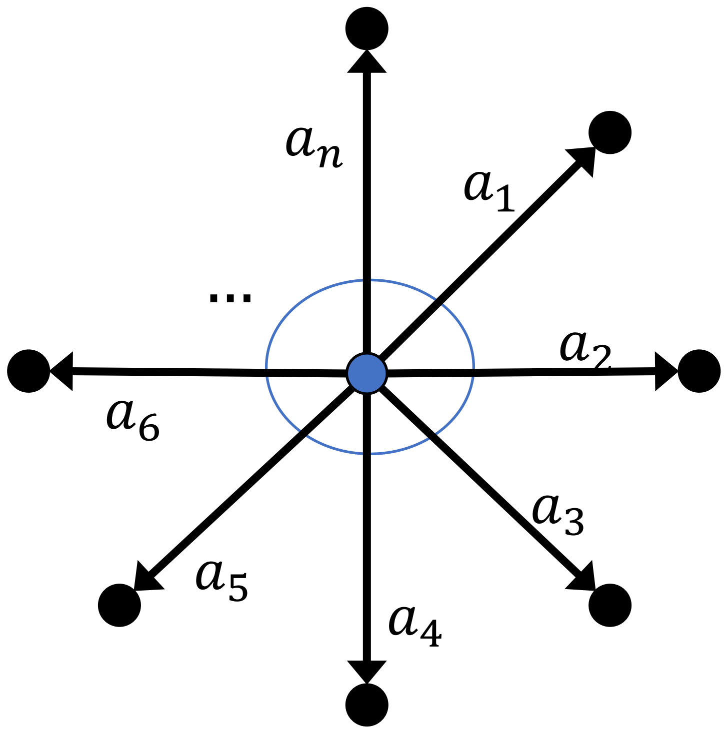
- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



A data structure called a priority queue allows finding the next nearest vertex in $O(\log n)$ time.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.

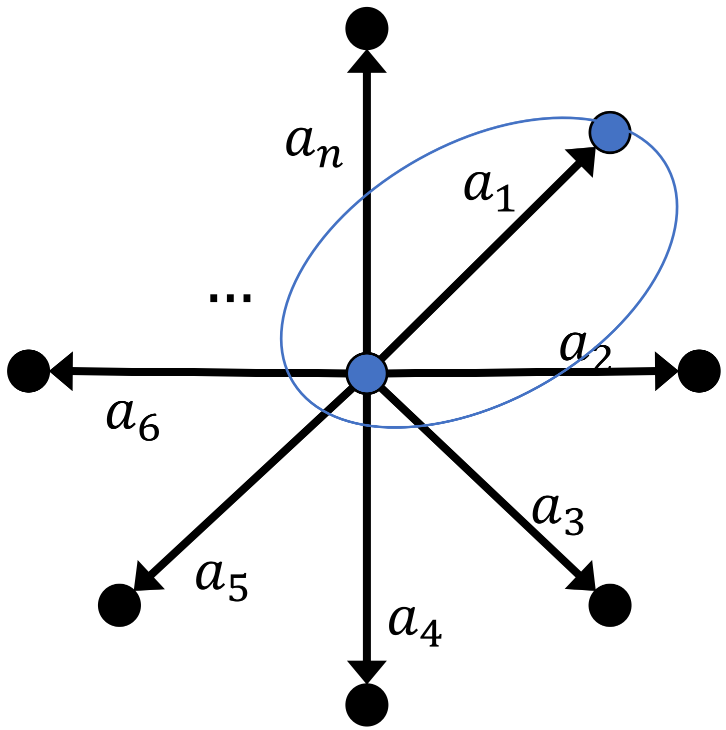


A data structure called a priority queue allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra's alg. visits vertices specifically in this order.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.

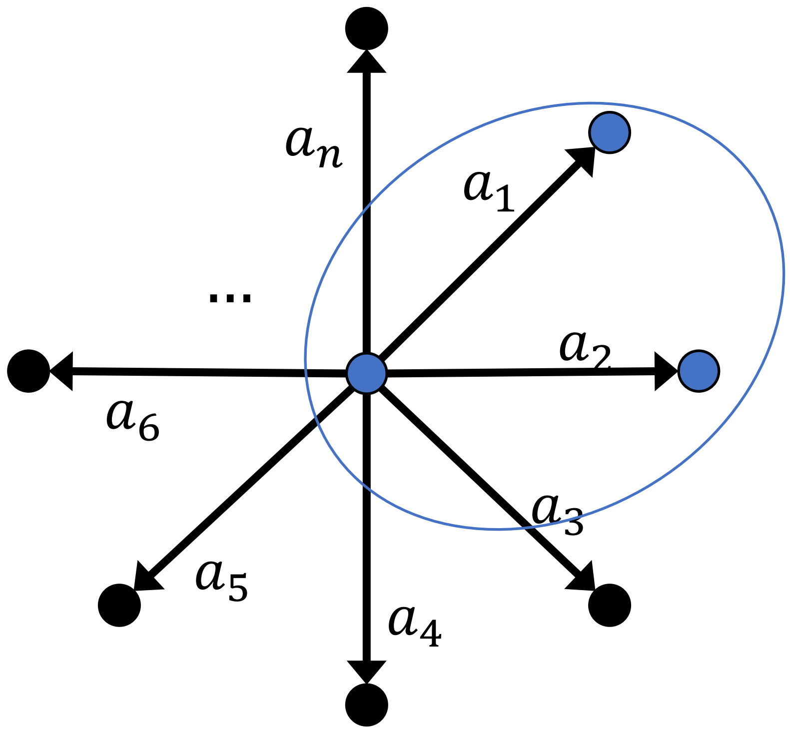


A data structure called a <u>priority queue</u> allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra's alg. visits vertices <u>specifically in this order</u>.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.

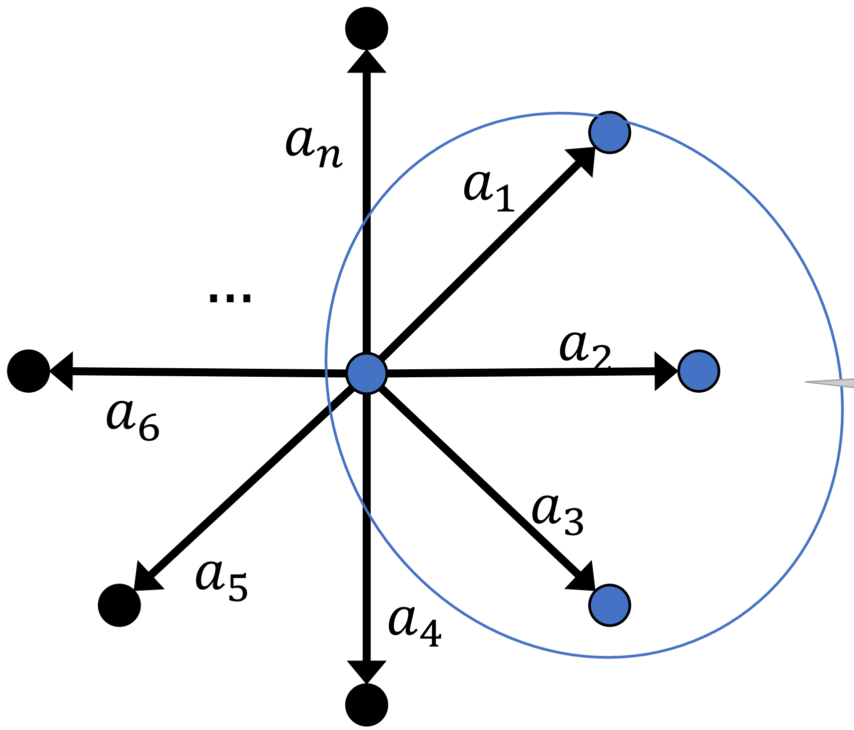


A data structure called a priority queue allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra's alg. visits vertices specifically in this order.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.

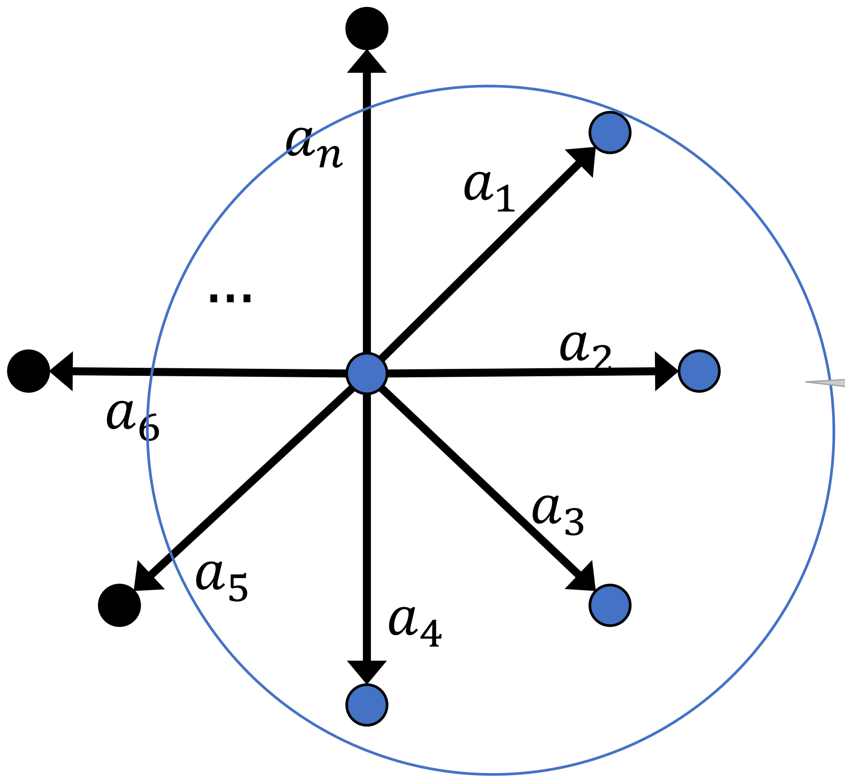


A data structure called a <u>priority queue</u> allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra's alg. visits vertices <u>specifically in this order</u>.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.

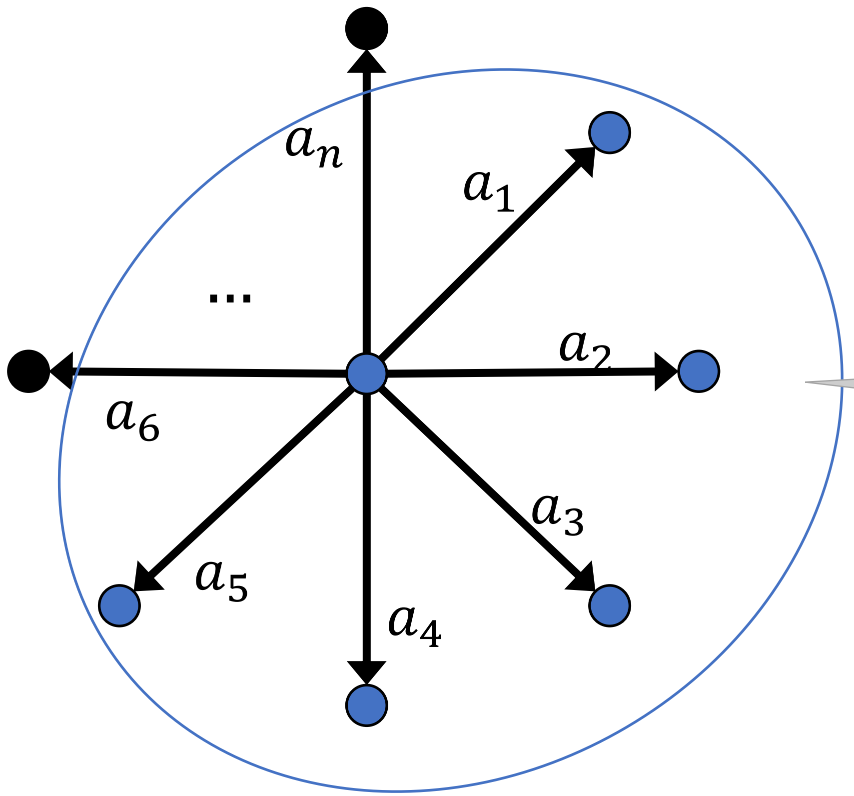


A data structure called a priority queue allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra's alg. visits vertices specifically in this order.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.

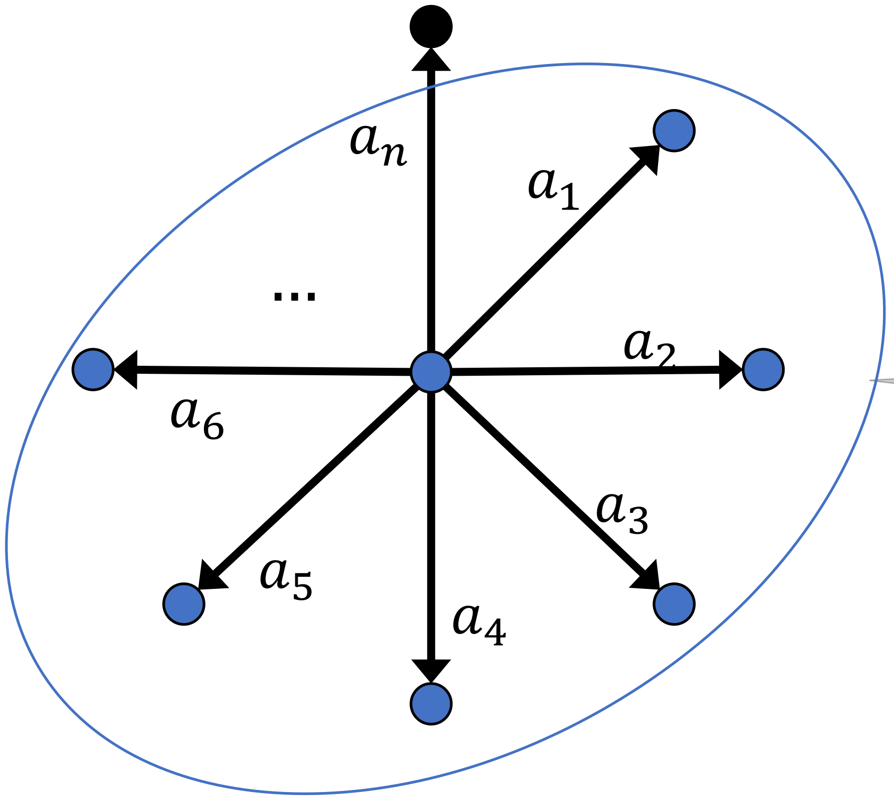


A data structure called a <u>priority queue</u> allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra’s alg. visits vertices <u>specifically in this order</u>.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge" from S.

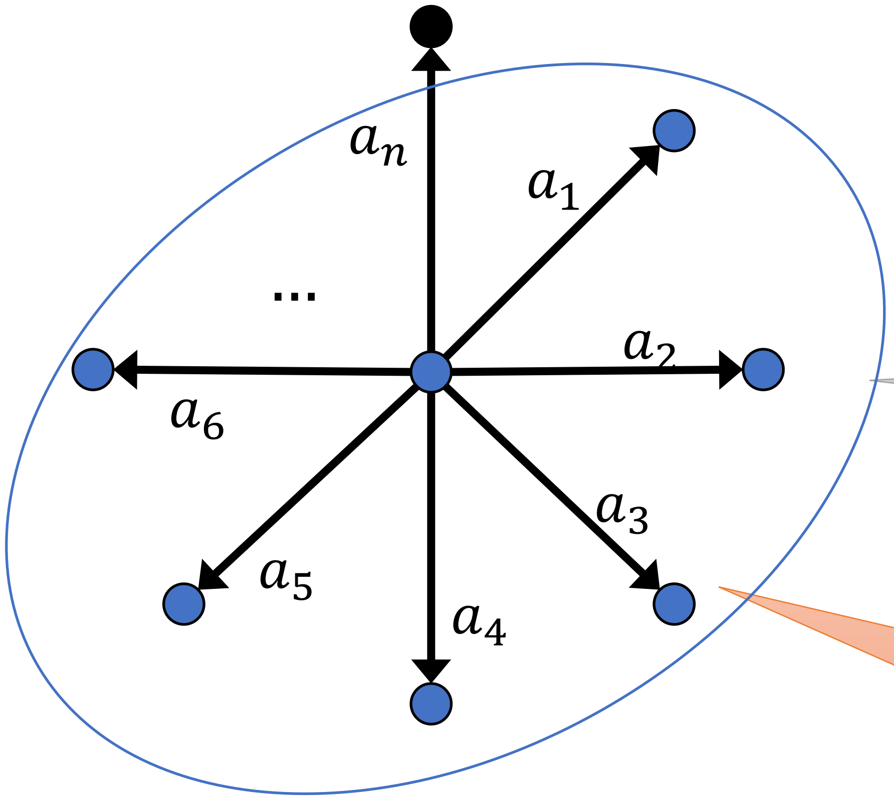


A data structure called a <u>priority queue</u> allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra's alg. visits vertices <u>specifically in this order</u>.

Dijkstra recap (non-negative weights)

- Suppose we have established correct distances to the k nearest vertices $S \subseteq V$.
- Then the $(k+1)$-th nearest vertex is the one „closest via a single edge” from S.



A data structure called a priority queue allows finding the next nearest vertex in $O(\log n)$ time.

If $a_1 < a_2 < \cdots < a_n$, Dijkstra's alg. visits vertices specifically in this order.

Sorting with binary comparisons requires $\log_2 n! = \Omega(n \log n)$ comparisons.

Recent breakthroughs

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
   "Negative-Weight Single-Source Shortest Paths in Near-linear Time"
2) STOC 2024 Best Paper Award (Fineman):
   "Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ *Time"*
3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
   "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Recent breakthroughs

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
   "Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) STOC 2024 Best Paper Award (Fineman):
   "Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ *Time"*

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
   "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Recent breakthroughs

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
   "Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) STOC 2024 Best Paper Award (Fineman):
   "Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}\left(mn^{8/9}\right)$ *Time"*

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
   "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Non-negative real weights.

Recent breakthroughs

Integer weights.

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
"Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) STOC 2024 Best Paper Award (Fineman):
"Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ *Time"*

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
"Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Non-negative real weights.

Recent breakthroughs

Integer weights.

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
"Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) STOC 2024 Best Paper Award (Fineman):
"Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ *Time"*

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
"Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Non-negative real weights.

➢ Real/integer regimes very different for SSSP.

Recent breakthroughs

Integer weights.

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
"Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) STOC 2024 Best Paper Award (Fineman):
"Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ Time"

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
"Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Non-negative real weights.

- Real/integer regimes very different for SSSP.
- What does it even mean to solve optimization problems on real numbers?

Exact optimization on real-weighted graphs

Real RAM:

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations ($+$, $-$, $\cdot$, $\div$) and comparisons performed in $O(1)$ time in a black-box way.

Exact optimization on real-weighted graphs

Real RAM:

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations ($+, -, \cdot, \div$) and comparisons performed in $O(1)$ time in a black-box way.

✓ Super convenient.

Exact optimization on real-weighted graphs

Real RAM:

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations ($+$, $-$, $\cdot$, $\div$) and comparisons performed in $O(1)$ time in a black-box way.

✓ Super convenient.

✓ Not very realistic.

Exact optimization on real-weighted graphs

Real RAM:

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations ($+$, $-$, $\cdot$, $\div$) and comparisons performed in $O(1)$ time in a black-box way.

✓ Super convenient.

✓ Not very realistic.

✓ Overpowered, e.g.:

- Integer division $\lfloor x/a \rfloor$ in $O(1)$ time $\Rightarrow$ solve NP-hard problems in poly. time.

Exact optimization on real-weighted graphs

Real RAM:

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations $(+, -, \cdot, \div)$ and comparisons performed in $O(1)$ time in a black-box way.

✓ Super convenient.

✓ Not very realistic.

✓ Overpowered, e.g.:

- Integer division $\lfloor x/a \rfloor$ in $O(1)$ time ⇒ solve NP-hard problems in poly. time.
- Only $(+, <)$ ⇒ e.g., count triangles in a graph in $\tilde{O}(m)$ time.

Exact optimization on real-weighted graphs

<u>Real RAM:</u>

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations ($+$, $-$, $\cdot$, $\div$) and comparisons performed in $O(1)$ time in a black-box way.

- Still, likely the best model for truly exact computation we've got…
- … if we do not abuse it. E.g. we can:
  - promise to only ever compute values from a natural restricted domain.

Exact optimization on real-weighted graphs

Real RAM:

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations ($+$, $-$, $\cdot$, $\div$) and comparisons performed in $O(1)$ time in a black-box way.

- Still, likely the best model for truly exact computation we've got…
- … if we do not abuse it. E.g. we can:
  - promise to only ever compute values from a natural restricted domain.
  - promise to use polynomial space if mapped to a realistic model.

Exact optimization on real-weighted graphs

Real RAM:

- Integer-indexed memory cells store infinite-precision real numbers,
- basic arithmetic operations ($+, -, \cdot, \div$) and comparisons performed in $O(1)$ time in a black-box way.

- Still, likely the best model for truly exact computation we've got…
- … if we do not abuse it. E.g. we can:
  - promise to only ever compute values from a natural restricted domain.
  - promise to use polynomial space if mapped to a realistic model.
- Green flag: running within the same time bound on a realistic model.

Recent breakthroughs (real weights)

1) <u>FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):</u>
   "Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) <u>STOC 2024 Best Paper Award (Fineman):</u>
   "Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ Time"

3) <u>STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):</u>
   "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Non-negative.

In both, all intermediate reals constructed are the graph's path lengths ⇒ no abuse!

Recent breakthroughs (real weights)

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
   "Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) STOC 2024 Best Paper Award (Fineman):
   "Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ Time"

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
   "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

Non-negative.

Runs in $O(m \log^{2/3} n)$ time using $(+, <)$, thus cannot sort via comparisons!

Baseline: $O(m + n \log n)$

In both, all intermediate reals constructed are the graph's path lengths ⇒ no abuse!

Word RAM

Think C language, $w = 64$.
$n =$ "problem size"

Word RAM:

- Memory cells store w-bit integers, where $w = \Omega(\log n)$ is the word size.
- Arithmetic/bitwise/comparison operations on w-bit integers performed in $O(1)$ time.

Word RAM

Think C language, $w = 64$.
$n =$ "problem size"

Word RAM:

- Memory cells store w-bit integers, where $w = \Omega(\log n)$ is the word size.
- Arithmetic/bitwise/comparison operations on w-bit integers performed in $O(1)$ time.

- Intermediate values sums of $\mathrm{poly}(n)$ input values $\rightarrow$ still $O(w)$-bit ints.

Word RAM

Think C language, $w = 64$.
$n =$ "problem size"

Word RAM:

- Memory cells store w-bit integers, where $w = \Omega(\log n)$ is the word size.
- Arithmetic/bitwise/comparison operations on w-bit integers performed in $O(1)$ time.

- Intermediate values sums of $\text{poly}(n)$ input values → still $O(w)$-bit ints.
- Typically, for integer weights fitting in a word, real RAM bounds transfer to word RAM bounds.

Word RAM

Think C language, $w = 64$.
$n =$ "problem size"

Word RAM:

- Memory cells store w-bit integers, where $w = \Omega(\log n)$ is the word size.
- Arithmetic/bitwise/comparison operations on w-bit integers performed in $O(1)$ time.

- Intermediate values sums of $\text{poly}(n)$ input values → still $O(w)$-bit ints.
- Typically, for integer weights fitting in a word, real RAM bounds transfer to word RAM bounds.
- Literature: "exact" means "exact on integer input".
  - For graphs: edge weights = integers fitting in a single word.

Word RAM

Think C language, $w = 64$.
$n =$ "problem size"

Word RAM:

- Memory cells store w-bit integers, where $w = \Omega(\log n)$ is the word size.
- Arithmetic/bitwise/comparison operations on w-bit integers performed in $O(1)$ time.

- Intermediate values sums of $\text{poly}(n)$ input values → still $O(w)$-bit ints.
- Typically, for integer weights fitting in a word, real RAM bounds transfer to word RAM bounds.
- Literature: "exact" means "exact on integer input".
  - For graphs: edge weights = integers fitting in a single word.

For simplicity, let's assume $w = \Theta(\log n) \Rightarrow$ absolute edge weights $\leq \text{poly}(n)$.

(obsolete) Manual for solving integer SSSP

Single-Source Shortest Paths (SSSP): [integer]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are integers fitting in words, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.

Does G have only non-negative weights?

(obsolete) Manual for solving integer SSSP

Single-Source Shortest Paths (SSSP): [integer]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are integers fitting in words, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.

Does G have only non-negative weights?

Yes? Use Dijkstra's algorithm with a fancier priority queue. $O(m + n \log\log n)$ time [Thorup '03].

Or faster; holds even for word size $\gg \log n$!

(obsolete) Manual for solving integer SSSP

Single-Source Shortest Paths (SSSP): [integer]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are integers fitting in words, and a source $s \in V$, compute $\text{dist}_G(s,t)$ for all $t \in V$.

Does G have only non-negative weights?

Yes? Use Dijkstra's algorithm with a fancier priority queue. $O(m + n \log\log n)$ time [Thorup '03].

Or faster; holds even for word size $\gg \log n$!

No? Use a "scaling" algorithm. E.g. Gabow's $\tilde{O}(mn^{3/4})$ ('83) Or Goldberg's $\tilde{O}(m\sqrt{n})$ ('93)

Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.

Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) \coloneqq \text{dist}_G(s, u)$ is one such function…

Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) \coloneqq \mathrm{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq 0.$$

Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) \coloneqq \text{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq 0.$$

1) does not change the shortest paths structure,

Reweighting via Price functions

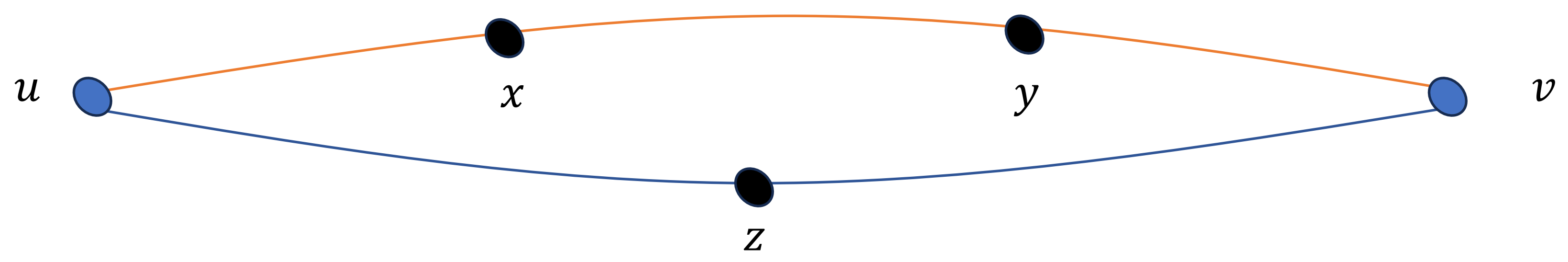
- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) \coloneqq \text{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq 0.$$

1) does not change the shortest paths structure,



Reweighting via Price functions

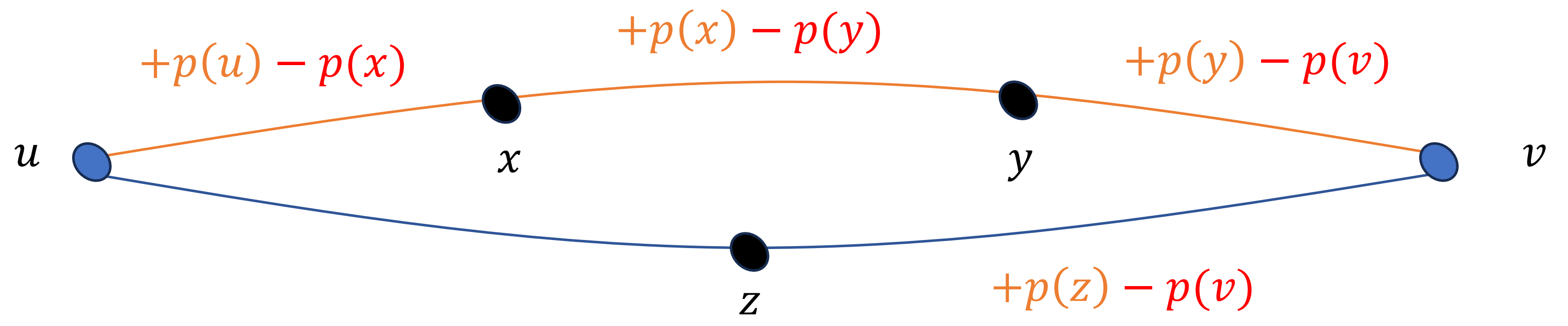
- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) := \text{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) := w(uv) + p(u) - p(v) \geq 0.$$

1) does not change the shortest paths structure,



Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

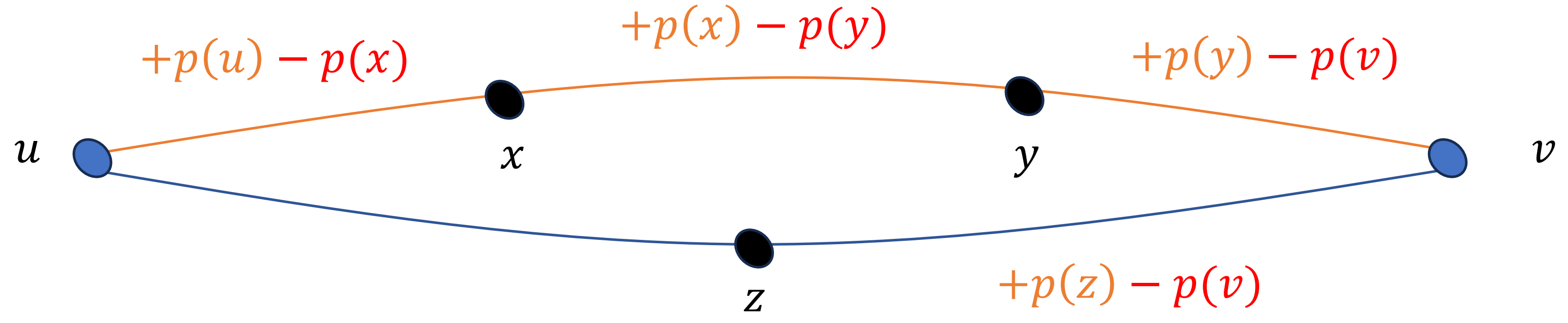
$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) := \mathrm{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) := w(uv) + p(u) - p(v) \geq 0.$$

1) does not change the shortest paths structure,

$$w_p(Q) = w(Q) + p(u) - p(v)$$



Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

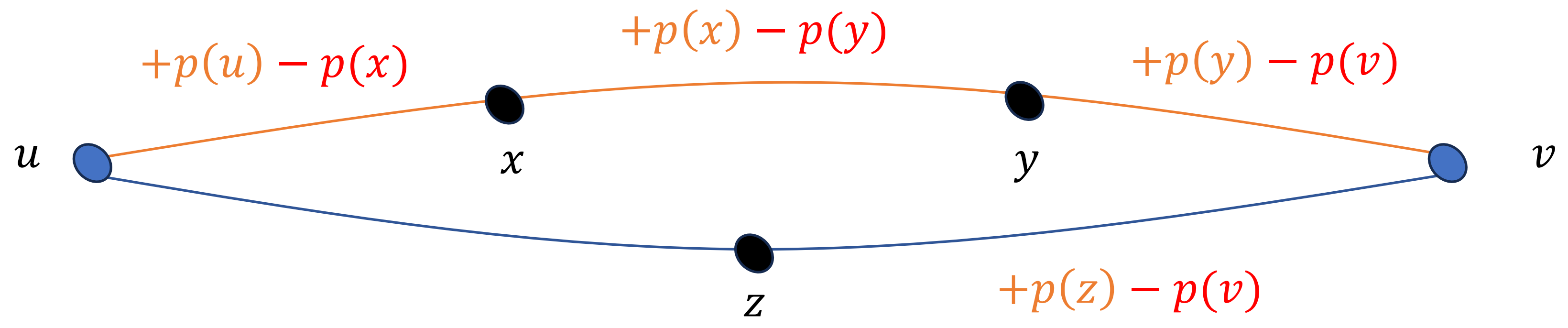
$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) \coloneqq \mathrm{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq 0.$$

1) does not change the shortest paths structure,

$$w_p(Q) = w(Q) + p(u) - p(v)$$
$$w_p(R) = w(R) + p(u) - p(v)$$



Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

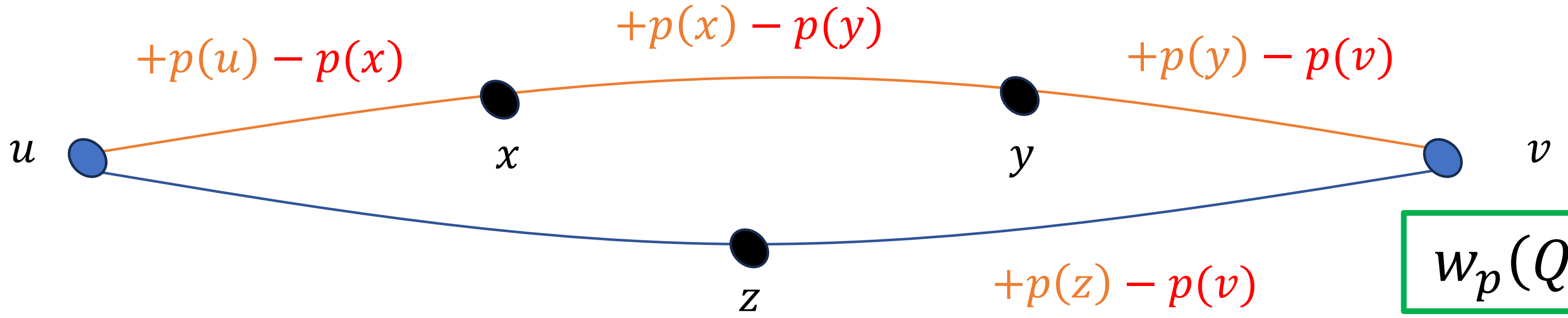
- $p(u) := \text{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) := w(uv) + p(u) - p(v) \geq 0.$$

1) does not change the shortest paths structure,

$$w_p(Q) = w(Q) + p(u) - p(v)$$

$$w_p(R) = w(R) + p(u) - p(v)$$



$$w_p(Q) - w_p(R) = w(Q) - w(R)$$

Reweighting via Price functions

- Consider vertex prices $p(v)$ for all $v \in V$.
- One can prove that for a well-defined SSSP problem there exist prices such that:

$$w(uv) + p(u) - p(v) \geq 0 \quad \text{for all edges } uv \in E$$

- $p(u) \coloneqq \mathrm{dist}_G(s, u)$ is one such function…
- Changing edge weights to:

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq 0.$$

1) does not change the shortest paths structure,
2) makes the problem amenable to Dijkstra.

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$

$|w(uv)| \leq \text{poly}(n)$, so start with $p \equiv 0$, $\epsilon = \text{poly}(n)$.

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

$$w(uv) + p(u) - p(v) \geq -\epsilon \quad \text{for all edges } uv \in E$$

2) Devise a refinement procedure that improves p to p' such that

$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \quad \text{for all edges } uv \in E$$

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$

2) Devise a refinement procedure that improves p to p' such that

$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \qquad \text{for all edges } uv \in E$$

3) After $O(\log n)$ iterations we will have

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq -\frac{1}{2n}.$$

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

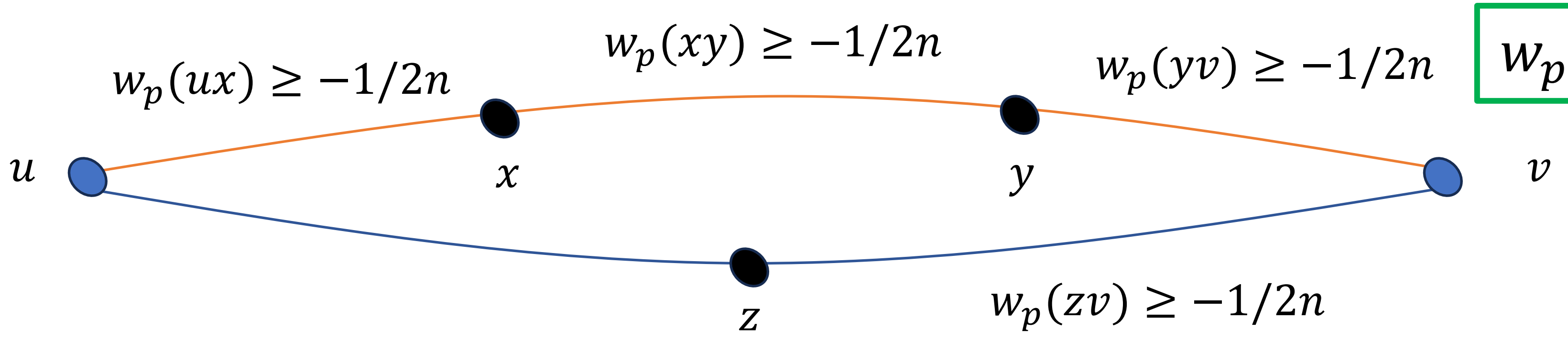
$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$

2) Devise a refinement procedure that improves p to p' such that

$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \qquad \text{for all edges } uv \in E$$

3) After $O(\log n)$ iterations we will have

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq -\frac{1}{2n}.$$



$$w_p(Q) - w_p(R) = w(Q) - w(R)$$

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

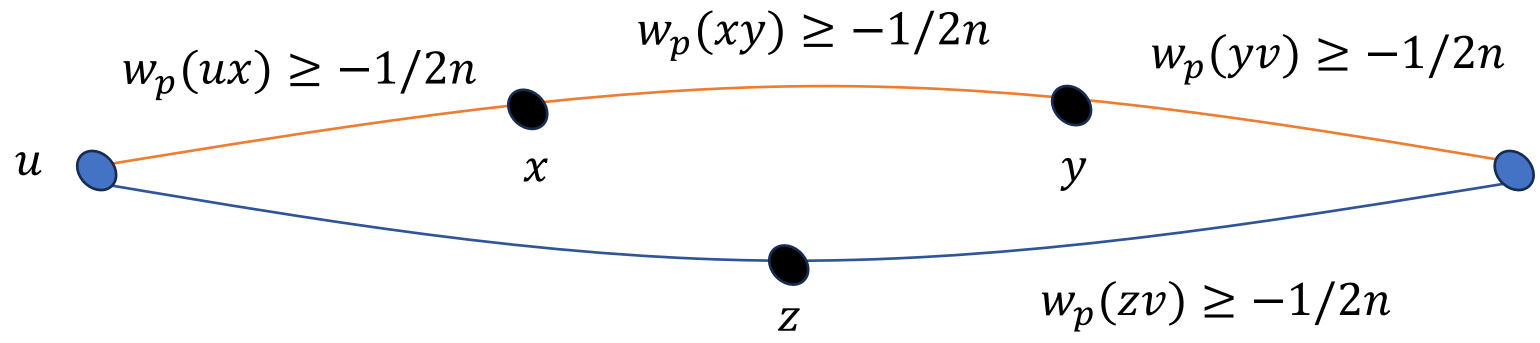
$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$

2) Devise a refinement procedure that improves p to p' such that

$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \qquad \text{for all edges } uv \in E$$

3) After $O(\log n)$ iterations we will have

$$w_p(uv) := w(uv) + p(u) - p(v) \geq -\frac{1}{2n}.$$

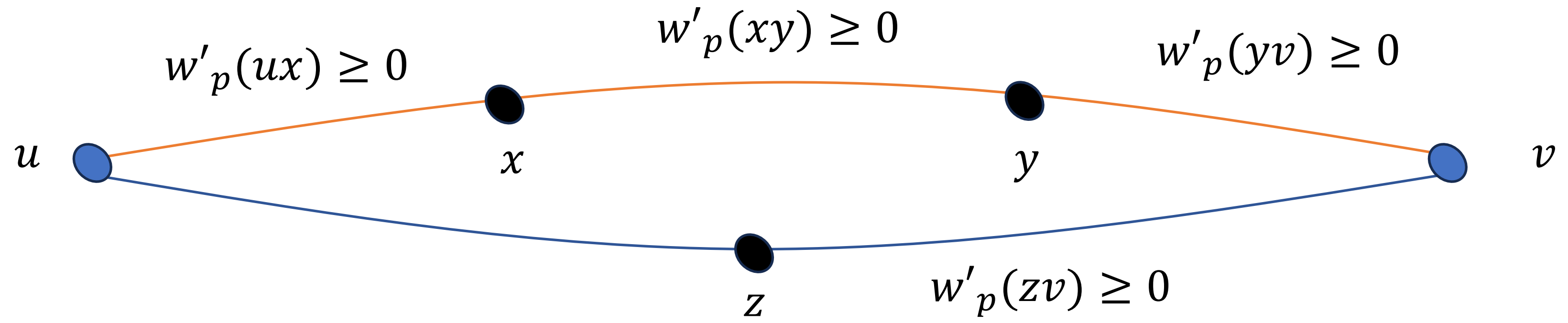


$w_p(Q) - w_p(R) = w(Q) - w(R)$

Increase weights by $\frac{1}{2}n$.

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily
$$w(uv) + p(u) - p(v) \geq -\epsilon \quad \text{for all edges } uv \in E$$
2) Devise a refinement procedure that improves p to p' such that
$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \quad \text{for all edges } uv \in E$$
3) After $O(\log n)$ iterations we will have
$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq -\frac{1}{2n}.$$



Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

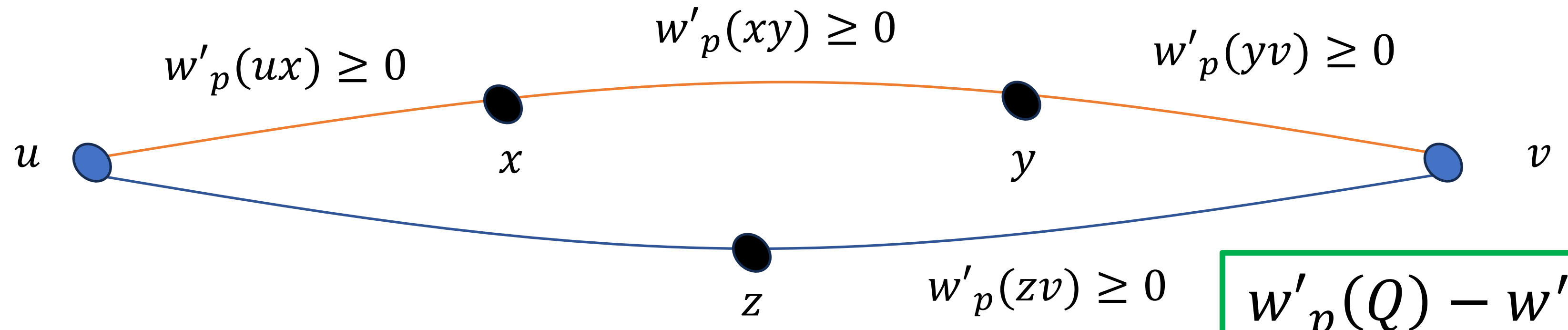
$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$

2) Devise a refinement procedure that improves p to p' such that

$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \qquad \text{for all edges } uv \in E$$

3) After $O(\log n)$ iterations we will have

$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq -\frac{1}{2n}.$$



$$w'_p(Q) - w'_p(R) \leq w(Q) - w(R) + 1/2$$

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily
$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$
2) Devise a refinement procedure that improves p to p' such that
$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \quad \text{for all edges } uv \in E$$
3) After $O(\log n)$ iterations we will have
$$w_p(uv) \coloneqq w(uv) + p(u) - p(v) \geq -\frac{1}{2n}.$$
4) Increasing edge weights by $\frac{1}{2n}$ increases path weight by at most $\frac{1}{2}$.
5) But, by integrality, shortest and non-shortest path weights differ by ≥ 1.
6) Dijkstra applicable and correct after increase.

Approximation scheme

1) Relax the inequalities with an error parameter ϵ that you can initialize easily

$$w(uv) + p(u) - p(v) \geq -\epsilon \qquad \text{for all edges } uv \in E$$

2) Devise a refinement procedure that improves p to p' such that

$$w(uv) + p'(u) - p'(v) \geq -\epsilon/2 \qquad \text{for all edges } uv \in E$$

3) After $O(\log(\text{poly}(n)))$ iterations we will have

$$w_p(uv) := w(uv) + p(u) - p(v) \geq -\frac{1}{2n}.$$

4) Increasing edge weights by $\frac{1}{2n}$ increases path weight by at most $\frac{1}{2}$.

5) But, by integrality, shortest and non-shortest path weights differ by ≥ 1.

6) Dijkstra applicable and correct after increase.

Integrality and scaling

Scaling framework = very efficient approximation scheme!

- Single ``refinement” iteration: decrease error by a constant factor.
- Additive error ϵ achieved in $\mathrm{polylog}(n, \epsilon^{-1})$ iterations.

Integrality and scaling

Scaling framework = very efficient approximation scheme!

- Single ``refinement” iteration: decrease error by a constant factor.
- Additive error ϵ achieved in $\mathrm{polylog}(n, \epsilon^{-1})$ iterations.

- Could be used even real data, e.g., irrational...
- ... but would never terminate with an exact solution in $\mathrm{poly}(n, m)$ time.

Integrality and scaling

Scaling framework = very efficient approximation scheme!

- Single ``refinement” iteration: decrease error by a constant factor.
- Additive error ϵ achieved in $\mathrm{polylog}(n, \epsilon^{-1})$ iterations.

- Could be used even real data, e.g., irrational...
- ... but would never terminate with an exact solution in $\mathrm{poly}(n, m)$ time.
- Integrality $\Rightarrow$ accuracy $\epsilon^{-1} = \mathrm{poly}(n)$ enough to correctly round.

Recent breakthroughs (integer)

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
“Negative-Weight Single-Source Shortest Paths in Near-linear Time”

2) STOC 2024 Best Paper Award (Fineman):
“Single-Source Shortest Paths with Nega

Approximation refinement in $\tilde{O}(m)$ time!

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
“Breaking the Sorting Barrier for Directed Single-Source Shortest Paths”

Recent breakthroughs (integer)

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
   “Negative-Weight Single-Source Shortest Paths in Near-linear Time”

2) STOC 2024 Best Paper Award (Fineman):
   “Single-Source Shortest Paths with Nega

Approximation refinement in $\tilde{O}(m)$ time!

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
   “Breaking the Sorting Barrier for Directed Single-Source Shortest Paths”

Other recent achievements:

- max flow with integer weights in $m^{1+o(1)}$ time [Chen et al. ‘22].

Recent breakthroughs (integer)

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
   “Negative-Weight Single-Source Shortest Paths in Near-linear Time”

2) STOC 2024 Best Paper Award (Fineman):
   “Single-Source Shortest Paths with Nega…”

3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
   “Breaking the Sorting Barrier for Directed Single-Source Shortest Paths”

Approximation refinement in $\tilde{O}(m)$ time!

Best real RAM bound:
$O(nm)$ [Orlin ‘13]

Other recent achievements:

➢ max flow with integer weights in $m^{1+o(1)}$ time [Chen et al. ‘22].

Recent breakthroughs (real weights)

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
"Negative-Weight Single-Source Shortest Paths in Near-linear Time"

2) STOC 2024 Best Paper Award (Fineman):
"Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}\left(mn^{8/9}\right)$ Time"

3) STOC 2025 Best Paper Awar … uan, Mao, Mao, Shu, Yin):
"Breaking the So … er for Directed Single-Source Shortest Paths"

Given a price function $p: V \to \mathbb{R}$ such that k vertices have adjacent negative edges, in $\tilde{O}\left(mk^{2/9}\right)$ *time one can compute a price fun.* p' *with* $k^{1/3}$ *fewer such negative vertices.*

Recent breakthroughs

1) FOCS 2022 Best Paper Award (Bernstein, Nanongkai, Wulff-Nilsen):
   "Negative-Weight Single-Source Shortest Paths in Near-linear Time"
2) STOC 2024 Best Paper Award (Fineman):
   "Single-Source Shortest Paths with Negative Real Weights in $\tilde{O}(mn^{8/9})$ *Time"*
3) STOC 2025 Best Paper Award (Duan, Mao, Mao, Shu, Yin):
   "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths"

What about rational data?

Exact optimization on graphs studied:

- either in an unrealistic real RAM model,
- or for integer data in a realistic word RAM model.

What about rational data?

Exact optimization on graphs studied:

- either in an unrealistic real RAM model,
- or for integer data in a realistic word RAM model.

✓ A rational number $\frac{p}{q}$ with $p, q = \Theta(\text{poly}(n))$ can be represented exactly using a single machine word in the word RAM.

What about rational data?

Exact optimization on graphs studied:

- either in an unrealistic real RAM model,
- or for integer data in a realistic word RAM model.

✓ A rational number $\frac{p}{q}$ with $p, q = \Theta(\mathrm{poly}(n))$ can be represented exactly using a single machine word in the word RAM.

Can optimization problems on rational-weighted graphs be solved on the word RAM **exactly** and as efficiently as on integer-weighted graphs?

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute $\text{dist}_G(s, t)$ for all $t \in V$.

- Suppose we adapt Dijkstra's algorithm: use exact arithmetic on rationals.

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute $\mathrm{dist}_G(s, t)$ for all $t \in V$.

- Suppose we adapt Dijkstra's algorithm: use exact arithmetic on rationals.
- Intermediate values = path lengths $\rightarrow$ sums of $O(n)$ word-fitting rationals.

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute $\text{dist}_G(s, t)$ for all $t \in V$.

- Suppose we adapt Dijkstra's algorithm: use exact arithmetic on rationals.
- Intermediate values = path lengths → sums of $O(n)$ word-fitting rationals.

$$\frac{1}{p_1} + \frac{1}{p_2} + \cdots + \frac{1}{p_k} = \frac{\text{something}}{p_1 p_2 \dots p_k}$$

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute ~~$\mathrm{dist}_G(s,t)$ for all $t \in V$~~.

- Suppose we adapt Dijkstra's algorithm: use exact arithmetic on rationals.
- Intermediate values = path lengths → sums of $O(n)$ word-fitting rationals.

$$\frac{1}{p_1} + \frac{1}{p_2} + \cdots + \frac{1}{p_k} = \frac{\text{something}}{p_1 p_2 \dots p_k}$$

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute a shortest paths tree from s.

- Suppose we adapt Dijkstra's algorithm: use exact arithmetic on rationals.
- Intermediate values = path lengths $\rightarrow$ sums of $O(n)$ word-fitting rationals.

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute a shortest paths tree from s.

- Suppose we adapt Dijkstra's algorithm: use exact arithmetic on rationals.
- Intermediate values = path lengths → sums of $O(n)$ word-fitting rationals.
- Intermediate values may require $\Omega(n)$ bits → arithmetic cost linear!

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute a shortest paths tree from s.

- Suppose we adapt Dijkstra's algorithm: use exact arithmetic on rationals.
- Intermediate values = path lengths → sums of $O(n)$ word-fitting rationals.
- Intermediate values may require $\Omega(n)$ bits → arithmetic cost linear!

Trivial adaptation of Dijkstra runs in $\tilde{O}(mn)$ time on rationals ≥ 0!

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute a shortest paths tree from s.

- Maybe apply some form of scaling then?

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute a shortest paths tree from s.

- Maybe apply some form of scaling then?
- But the second shortest path might differ from OPT by $2^{-\tilde{\Theta}(n)}$.

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute a shortest paths tree from s.

- Maybe apply some form of scaling then?
- But the second shortest path might differ from OPT by $2^{-\tilde{\Theta}(n)}$.
- Scaling needs accuracy $\epsilon = 2^{-\tilde{\Theta}(n)}$.

Rational SSSP

Single-Source Shortest Paths (SSSP): [rational]
Given a directed graph $G = (V, E)$ with n vertices and m edges whose weights are word-fitting rationals, and a source $s \in V$, compute a shortest paths tree from s.

- Maybe apply some form of scaling then?
- But the second shortest path might differ from OPT by $2^{-\widetilde{\Theta}(n)}$.
- Scaling needs accuracy $\epsilon = 2^{-\widetilde{\Theta}(n)}$.

At least $\Theta(n)$-factor time slowdown compared to integer data!

Our results

Joint work with W. Nadara and M. Sokołowski (SODA 2024):

Theorem:
SSSP with non-negative word-fitting rational weights can be solved in $\tilde{O}(n+m)$ time on the word RAM.

Our results

Joint work with W. Nadara and M. Sokołowski (SODA 2024):

Theorem:
SSSP with non-negative word-fitting rational weights can be solved in $\tilde{O}(n+m)$ time on the word RAM.

- … even though arithmetic operations on k-bit rationals take $\tilde{O}(k)$ time.

Our results

Joint work with W. Nadara and M. Sokołowski (SODA 2024):

Theorem:
SSSP with non-negative word-fitting rational weights can be solved in $\tilde{O}(n+m)$ time on the word RAM.

- … even though arithmetic operations on k-bit rationals take $\tilde{O}(k)$ time.
- Indeed, almost matching the best-known integer bound possible for $\mathrm{SSSP}_{\geq 0}$.

Our results

Joint work with W. Nadara and M. Sokołowski (SODA 2024):

Theorem:
SSSP with word-fitting rational weights can be solved in $\tilde{O}(m + n^{2.5})$ time on the word RAM.

Our results

Joint work with W. Nadara and M. Sokołowski (SODA 2024):

<u>Theorem:</u>
SSSP with word-fitting rational weights can be solved in $\tilde{O}(m + n^{2.5})$ time on the word RAM.

- Beats scaling with exponential accuracy for very dense graphs $m = \Omega(n^{2.51})$.

Our results

Joint work with W. Nadara and M. Sokołowski (SODA 2024):

Theorem:
SSSP with word-fitting rational weights can be solved in $\tilde{O}(m + n^{2.5})$ time on the word RAM.

- Beats scaling with exponential accuracy for very dense graphs $m = \Omega(n^{2.51})$.
- No reason to believe near-linear time is impossible.

Conclusion

1) Model choice and low-level details can make a huge difference even for very basic polynomial algorithmic optimization problems.

Conclusion

1) Model choice and low-level details can make a huge difference even for very basic polynomial algorithmic optimization problems.
2) What we’ve been taught about computing single-source shortest paths is now completely obsolete at last.

Conclusion

1) Model choice and low-level details can make a huge difference even for very basic polynomial algorithmic optimization problems.

2) What we've been taught about computing single-source shortest paths is now completely obsolete at last.

3) Very fast-converging approximation schemes can be considered exact algorithms in realistic models of computation.

Conclusion

1) Model choice and low-level details can make a huge difference even for very basic polynomial algorithmic optimization problems.
2) What we've been taught about computing single-source shortest paths is now completely obsolete at last.
3) Very fast-converging approximation schemes can be considered exact algorithms in realistic models of computation.
4) Studying truly exact computation in unrealistic models is okay and timely, but don't abuse them!

Open problem

See https://en.wikipedia.org/wiki/Smale's_problems

Big open problem in Real RAM vs. Word RAM optimization:

Can Linear Programming be solved exactly on a Real RAM in polynomial time (as a function of $\#(\text{variables} + \text{constraints})$) without model abuse?

Open problem

See https://en.wikipedia.org/wiki/Smale's_problems

Big open problem in Real RAM vs. Word RAM optimization:

Can Linear Programming be solved exactly on a Real RAM in polynomial time (as a function of $\#(\text{variables} + \text{constraints})$) without model abuse?

Khachiyan'79: Linear programming with rational data can be solved in polynomial time (in $\#(\text{variables} + \text{constraints})$, on the word RAM).