Fast Shortest-Path Queries

Laurent Viennot

MPRI - Theory of practical graph algorithms
Fast shortest-path queries
Shortest path queries

Problem:
• A graph $G$ is given.
• Answer queries: shortest path from $s$ to $t$?

Trivial solution: pre-compute for all $s, t$.

Recent progress [BDG+15], e.g. in road networks ($n = 20M$):
• Dijkstra: 4s
• Bidirectional Dijkstra: 1s
• Bidirectional A*: 100ms
• Reach-Pruning, Contraction Hierarchies: 10 ms
• Hub labeling: 10 µs
Shortest path queries

**Problem:**
- A graph $G$ is given.
- Answer queries: shortest path from $s$ to $t$?

**Trivial solution:** pre-compute for all $s, t$.

**Recent progress** [BDG+15], e.g. in road networks ($n = 20M$):
- Dijkstra: 4s
- Bidirectional Dijkstra: 1s
- Bidirectional A*: 100ms
- Reach-Pruning, Contraction Hierarchies: 10 ms
- Hub labeling: 10 μs
Shortest path queries

Problem:
- A graph $G$ is given.
- Answer queries: shortest path from $s$ to $t$?

Trivial solution: pre-compute for all $s, t$.

Recent progress [BDG+15], e.g. in road networks ($n = 20M$):
- Dijkstra: 4s
- Bidirectional Dijkstra: 1s
- Bidirectional A*: 100ms
- Reach-Pruning, Contraction Hierarchies: 10 ms
- Hub labeling: 10 $\mu$s
Shortest path queries

Problem:

- A graph $G$ is given.
- Answer queries: shortest path from $s$ to $t$?

Trivial solution: pre-compute for all $s, t$.

Recent progress [BDG+15], e.g. in road networks ($n = 20M$):

- Dijkstra: 4s
- Bidirectional Dijkstra: 1s
- Bidirectional A*: 100ms
- Reach-Pruning, Contraction Hierarchies: 10 ms
- Hub labeling: 10 $\mu$s
Shortest path queries

Problem:
• A graph $G$ is given.
• Answer queries: shortest path from $s$ to $t$?

Trivial solution: pre-compute for all $s, t$.

Recent progress [BDG+15], e.g. in road networks ($n = 20M$):
• Dijkstra: 4s
• Bidirectional Dijkstra: 1s
• Bidirectional $A^*$: 100ms
• Reach-Pruning, Contraction Hierarchies: 10 ms
• Hub labeling: 10 $\mu$s
Shortest path queries

**Problem:**
- A graph $G$ is given.
- Answer queries: shortest path from $s$ to $t$?

**Trivial solution:** pre-compute for all $s, t$.

**Recent progress [BDG+15], e.g. in road networks ($n = 20M$):**
- Dijkstra: 4s
- Bidirectional Dijkstra: 1s
- Bidirectional $A^*$: 100ms
- Reach-Pruning, Contraction Hierarchies: 10 ms
- Hub labeling: 10 μs
Shortest path queries

Problem:

- A graph $G$ is given.
- Answer queries: shortest path from $s$ to $t$?

Trivial solution: pre-compute for all $s, t$.

Recent progress [BDG+15], e.g. in road networks $(n = 20M)$:

- Dijkstra: 4s
- Bidirectional Dijkstra: 1s
- Bidirectional A*: 100ms
- Reach-Pruning, Contraction Hierarchies: 10 ms
- Hub labeling: 10 μs
Shortest path queries

Problem:
- A graph $G$ is given.
- Answer queries: shortest path from $s$ to $t$?

Trivial solution: pre-compute for all $s, t$.

Recent progress [BDG+15], e.g. in road networks ($n = 20M$):
- Dijkstra: 4s
- Bidirectional Dijkstra: 1s
- Bidirectional $A^*$: 100ms
- Reach-Pruning, Contraction Hierarchies: 10 ms
- Hub labeling: 10 $\mu$s
Shortest path queries

Problem:
- A graph $G$ is given.
- Answer queries: shortest path from $s$ to $t$?

Trivial solution: pre-compute for all $s, t$.

Recent progress [BDG+15], e.g. in road networks ($n = 20M$):
- Dijkstra: 4s
- Bidirectional Dijkstra: 1s
- Bidirectional $A^*$: 100ms
- Reach-Pruning, Contraction Hierarchies: 10 ms
- Hub labeling: 10 $\mu$s
Figure 7. Preprocessing and average query time performance for algorithms with available experimental data on the road network of Western Europe, using travel times as edge weights. Connecting lines indicate different trade-offs for the same algorithm. The figure is inspired by [238].

Note that a machine with more than one petabyte of RAM (as required by this algorithm) would likely have slower memory access times. Times in the plot are on a single core of an Intel X5680 3.33 GHz CPU, a mainstream server at the time of writing. Several of the algorithms in the plot were originally run on this machine [5, 75, 77, 82]; for the remaining, we divide by the following scaling factors: 2.322 for [40, 83], 2.698 for [142], 1.568 for [15], 0.837 for [107], and 0.797 for [112]. These were obtained from a benchmark (developed for this survey) that measures the time of computing several shortest path trees on the publicly available USA road network with travel times [101]. For the machines we did not have access to, we asked the authors to run the benchmark for us [112]. The benchmark is available from http://algo.iti.kit.edu/~pajor/survey/.
Query time vs space (exact) [Sommer 14]

Fig. 4. Route planning for road networks: the tradeoff between space $S$ and query time $Q$ for recent shortest-path query data structures, depicted using doubly logarithmic scales. The performance numbers represented by this figure were extracted from Delling et al. [2009a, Table 1], Bauer et al. [2010b, Table 8], Abraham et al. [2011b, Table 1], Arz et al. [2013, arXiv Table 5], and Delling et al. [2013b, Table 2]. Performance numbers were obtained on different machines and scaled with best effort to make methods comparable. Colors and dashed lines do not carry any meaning; lines serve the purpose of visually connecting dots corresponding to different implementations or different variants of the same method.

Methods using contraction hierarchies (CH) [Geisberger et al. 2008; Sanders et al. 2008; Geisberger et al. 2012] dominate the low-space regime; methods based on reach [Gutman 2004; Goldberg et al. 2009] and highway hierarchies (HH) [Sanders and Schultes 2005, 2006; Delling et al. 2009b] can be seen as the “first generation” of CH; transit-node routing (TNR) [Bast et al. 2007a; Bauer et al. 2010b; Arz et al. 2013] and hub labels (HL) [Abraham et al. 2011b; Delling et al. 2013b] dominate the fast-query-time regime.

Efficient practical methods to answer shortest-path queries are often devised by following a feedback loop that consists of four steps: design, analysis, implementation, and experimentation. This approach is also called algorithm engineering [Sanders 2009, Figure 1]. Since experimentation is an integral part of the feedback loop, the choice of the datasets may highly influence the outcome of the algorithm engineering process. Whenever possible, experiments are run with input graphs that are actually used in practice. Route planning methods discovered by an algorithm engineering process include, for example, Highway Hierarchies (HH) [Sanders and Schultes 2005, 2006] and its exceedingly popular successor called Contraction Hierarchies (CH) [Geisberger et al. 2008]. Both methods depend on structural properties of the input graph and rather heavily on the edge lengths and the shortest-path metric they impose. If the length function is chosen such that edge lengths correspond to Euclidean distances, the methods still work well but their performance is worse than the performance when edge lengths correspond to (estimated) travel times. It is for the so-called travel time metric, where the hierarchical methods excel, and where the performance obtained are truly impressive (see also other methods, as illustrated in Figure 4). However, estimating travel times for road segments is a highly nontrivial task in itself and it is not entirely clear to what extent the estimates used in research datasets are accurate representations for actual travel times observed in the real world. To the best of my knowledge, there are only a few studies on the robustness of these methods, investigating whether the performances would drop significantly upon changes to the length function; see, for example, Delling et al. [2013a]. Recent theoretical research (Section 3.2.4) strives to explain the success of these speedup techniques, analyzing...
Distance oracles

**Def:** Given a graph $G$ with $n$ nodes, compute a data-structure of size $S$ allowing to answer queries “what is $d_G(u, v)$?” for $u, v \in V(G)$ in time $t$.

**Th:** [Cohen Porat 2010] $t = O(1)$ requires space $S = \Omega(n^2)$ under Fast-Set-Intersection Hypothesis (given $S_1, \ldots, S_n \in [\log^c n]$, answering queries “does $S_i$ intersects $S_j$?” in constant time requires $\Omega(n^2)$ space). This holds even for $2 - \varepsilon$-approximation.

**Best algorithm:** $O(t)$ time with $\left(\frac{n^2}{t^2}\right)$ space. [Cohen Porat 2010].
Distance oracles

**Def**: Given a graph $G$ with $n$ nodes, compute a data-structure of size $S$ allowing to answer queries “what is $d_G(u, v)$?” for $u, v \in V(G)$ in time $t$.

**Th**: [Cohen Porat 2010] $t = O(1)$ requires space $S = \Omega(n^2)$ under Fast-Set-Intersection Hypothesis (given $S_1, \ldots, S_n \in [\log^c n]$, answering queries “does $S_i$ intersects $S_j$?” in constant time requires $\Omega(n^2)$ space). This holds even for $2 - \varepsilon$-approximation.

**Best algorithm**: $O(t)$ time with $(n^2/t^2)$ space. [Cohen Porat 2010].
Distance oracles

**Def:** Given a graph $G$ with $n$ nodes, compute a data-structure of size $S$ allowing to answer queries “what is $d_G(u, v)$?” for $u, v \in V(G)$ in time $t$.

**Th:** [Cohen Porat 2010] $t = O(1)$ requires space $S = \Omega(n^2)$ under Fast-Set-Intersection Hypothesis (given $S_1, \ldots, S_n \in [\log^c n]$, answering queries “does $S_i$ intersects $S_j$?” in constant time requires $\Omega(n^2)$ space). This holds even for $2 - \varepsilon$-approximation.

**Best algorithm:** $O(t)$ time with $(n^2/t^2)$ space. [Cohen Porat 2010].
A short history of shortest paths
Dijkstra [Dijkstra ’59]

**Procedure** Dijkstra \((G, s, t)\)

- Distance label \(d(u) := 0\) if \(u = s\), \(\infty\).
- Radius \(r := 0\).

**Repeat**

- Pick unscanned \(u\) with \(d(u)\) min.  // \(d(u) = d_G(s, u)\)

**Scan** \(u\):

- For \(v \in N_G(u)\) do \(d(v) := \min\{d(v), d(u) + \ell(uv)\}\)
- \(r := \min_{\text{unscanned } v} d(v)\)

**Until** \(d(t) \leq r\)

**Return** \(d(t)\)
Bidirectional Dijkstra

Procedure BidirDijkstra \((G, s, t)\)

Alternate Dijkstra \((G, s, t)\) and Dijkstra \((\bar{G}, t, s)\).

Stopping condition?
Bidirectional Dijkstra

**Procedure** 

BidirDijkstra \((G, s, t)\)

- Alternate Dijkstra \((G, s, t)\) and Dijkstra \((\hat{G}, t, s)\).
- Stopping condition?

Estimation \(= \frac{1}{2}d(s, t)\).

When scanning edge \(uv\):

\[
\min \left\{d(u) + \ell(uv) + d(v)\right\}.
\]

Stop if \(r + \bar{r}\).
Bidirectional Dijkstra

**Procedure** BidirDijkstra \((G, s, t)\)

- Alternate Dijkstra \((G, s, t)\) and Dijkstra \((\overline{G}, t, s)\).
- Stopping condition?

Estimation \(\mu := \infty\) of \(d(s, t)\).

When scanning edge \(uv\) : \(\mu := \min \left\{ \mu, d(u) + \ell(uv) + d(v) \right\}\).
**Bidirectional Dijkstra**

**Procedure** BidirDijkstra \((G, s, t)\)

1. Alternate Dijkstra \((G, s, t)\) and Dijkstra \((\overline{G}, t, s)\).
2. Stopping condition?

**Estimation** \(\mu := \infty\) of \(d(s, t)\).

When scanning edge \(uv\):

\[
\mu := \min \left\{ \mu, d(u) + \ell(uv) + d(v) \right\}.
\]

Stop if \(\mu \leq r + \overline{r}\).
**Bidirectional Dijkstra**

**Procedure** `BidirDijkstra (G, s, t)`

Alternate Dijkstra `(G, s, t)` and Dijkstra `(G, t, s)`. 
Stopping condition? 

Estimation $\mu := \infty$ of $d(s, t)$.
When scanning edge $uv$: $\mu := \min \left\{ \mu, d(u) + \ell(uv) + d(v) \right\}$.
Stop if $\mu \leq r + \bar{r}$.

**Exercise**: show correctness.
Bidirectional Dijkstra

**Procedure** BidirDijkstra \((G, s, t)\)

- Alternate Dijkstra \((G, s, t)\) and Dijkstra \((\widehat{G}, t, s)\).
- Stopping condition?

Estimation \(\mu := \infty\) of \(d(s, t)\).

When scanning edge \(uv\): \(\mu := \min \left\{ \mu, d(u) + \ell(uv) + \widehat{d}(v) \right\}\).

Stop if \(\mu \leq r + \widehat{r}\).

**Exercise** : show correctness.

**Exercise** : explain 1s vs 4s in road networks.
A* [Hart, Nilsson, Raphael ’68]

Shortest path algorithm with prediction:

Potential function $\pi(u) \approx d(u, t)$.

Dijkstra ($G_\pi, s, t$) with $\ell_\pi(uv) = \ell(uv) - (\pi(u) - \pi(v))$.

- $d_\pi(s, t) = d(s, t) - (\pi(s) - \pi(t))$.
- Scan $u$ with $d(u) + \pi(u) \text{ min}$.

$\pi$ feasible if $\forall uv \in E(G), \ell_\pi(uv) \geq 0$

Exercise: Condition for using $\pi(u) = D(u, t)$ for a metric $D$?

Bidirectional A* : ALT [Goldberg, Harelsson ’05]

(ALT = A*, Landmarks, Triangle inequality)

- $\ell_\pi(uv) = \ell \leftarrow \pi(vu) \iff \pi + \frac{\ell_\pi}{\pi} = cte$
  
  (ex: $\pi' = (\pi - \frac{\ell_\pi}{\pi})/2$ and $\pi' = (\frac{\ell_\pi}{\pi} - \pi)/2$)

- or different stopping condition

- $\pi$ from landmarks (better than using coordinates):
  
  $\pi(u) = \max_{x \in X} d_G(u, x) - d_G(t, x)$

Exercise: Prove it works for bidir. A*.
A* [Hart, Nilsson, Raphael ’68]

Shortest path algorithm with prediction:

Potential function $\pi(u) \approx d(u, t)$.

Dijkstra ($G_\pi$, s, t) with $\ell_\pi(uv) = \ell(uv) - (\pi(u) - \pi(v))$.

- $d_\pi(s, t) = d(s, t) - (\pi(s) - \pi(t))$.
- Scan u with $d(u) + \pi(u)$ min.

$\pi$ feasible if $\forall uv \in E(G), \ell_\pi(uv) \geq 0$

Exercise: Condition for using $\pi(u) = D(u, t)$ for a metric D?

Bidirectional A* : ALT [Goldberg, Harelsson ’05]

(ALT = A*, Landmarks, Triangle inequality)

- $\ell_\pi(uv) = \ell_\pi(vu) \iff \pi + \ell_\pi = \text{cte}$
  (ex: $\pi' = (\pi - \ell_\pi)/2$ and $\pi' = (\ell_\pi - \pi)/2$)

- or different stopping condition
- $\pi$ from landmarks (better than using coordinates):
  $\pi(u) = \max_{x \in X} d_G(u, x) - d_G(t, x)$

Exercise: Prove it works for bidir. A*.
A* [Hart, Nilsson, Raphael ’68]

Shortest path algorithm with prediction:

Potential function $\pi(u) \approx d(u, t)$.

Dijkstra ($G_\pi, s, t$) with $\ell_\pi(uv) = \ell(uv) - (\pi(u) - \pi(v))$.

$d_\pi(s, t) = d(s, t) - (\pi(s) - \pi(t))$.

Scan $u$ with $d(u) + \pi(u)$ min.

$\pi$ feasible if $\forall uv \in E(G), \ell_\pi(uv) \geq 0$

Exercise: Condition for using $\pi(u) = D(u, t)$ for a metric $D$?

Bidirectional $A^*$: ALT [Goldberg, Harelsson ’05]

(ALT = $A^*$, Landmarks, Triangle inequality)

• $\ell_\pi(uv) = \ell_\pi(vu) \iff \pi + \ell_\pi = \text{cte}$
  (ex: $\pi' = (\pi - \ell_\pi)/2$ and $\pi' = (\ell_\pi - \pi)/2$)

• or different stopping condition

• $\pi$ from landmarks (better than using coordinates):

  $\pi(u) = \max_{x \in X} d_G(u, x) - d_G(t, x)$

Exercise: Prove it works for bidir. $A^*$.
A* [Hart, Nilsson, Raphael ’68]

Shortest path algorithm with prediction:

Potential function $\pi(u) \approx d(u, t)$.
Dijkstra $(G, s, t)$ with $\ell_\pi(uv) = \ell(uv) - (\pi(u) - \pi(v))$.

$d_\pi(s, t) = d(s, t) - (\pi(s) - \pi(t))$.

Scan $u$ with $d(u) + \pi(u)$ min.

$\pi$ feasible if $\forall uv \in E(G), \ell_\pi(uv) \geq 0$

Exercise: Condition for using $\pi(u) = D(u, t)$ for a metric $D$?

Bidirectional A* : ALT [Goldberg, Harelsson ’05]
(ALT = A*, Landmarks, Triangle inequality)

- $\ell_\pi(uv) = \ell_\pi(vu) \iff \pi + \ell_\pi = \text{cte}$
  (ex: $\pi' = (\pi - \ell_\pi)/2$ and $\pi' = (\ell_\pi - \pi)/2$)

- or different stopping condition

- $\pi$ from landmarks (better than using coordinates):
  
  $\pi(u) = \max_{x \in X} d_G(u, x) - d_G(t, x)$

Exercise: Prove it works for bidir. A*.
A* [Hart, Nilsson, Raphael '68]

Shortest path algorithm with prediction:

Potential function $\pi(u) \approx d(u, t)$.

Dijkstra $(G_\pi, s, t)$ with $\ell_\pi(uv) = \ell(uv) - (\pi(u) - \pi(v))$.

$d_\pi(s, t) = d(s, t) - (\pi(s) - \pi(t))$.

Scan $u$ with $d(u) + \pi(u)$ min.

$\pi$ feasible if $\forall uv \in E(G), \ell_\pi(uv) \geq 0$

Exercise: Condition for using $\pi(u) = D(u, t)$ for a metric $D$?

Bidirectional A* : ALT [Goldberg, Harelsson ’05]

(ALT = A*, Landmarks, Triangle inequality)

- $\ell_\pi(uv) = \ell_{\pi}(vu) \iff \pi + \bar{\pi} = cte$
  (ex: $\pi' = (\pi - \bar{\pi})/2$ and $\pi' = (\bar{\pi} - \pi)/2$)

- or different stopping condition

- $\pi$ from landmarks (better than using coordinates):
  $\pi(u) = \max_{x \in X} d_G(u, x) - d_G(t, x)$

Exercise: Prove it works for bidir. A*.
A* [Hart, Nilsson, Raphael ‘68]

Shortest path algorithm with prediction:

Potential function $\pi(u) \approx d(u, t)$.

Dijkstra ($G_\pi, s, t$) with $\ell_\pi(uv) = \ell(uv) - (\pi(u) - \pi(v))$.

$\begin{align*}
d_\pi(s, t) &= d(s, t) - (\pi(s) - \pi(t)). \\
\end{align*}$

Scan $u$ with $d(u) + \pi(u)$ min.

$\pi$ feasible if $\forall uv \in E(G), \ell_\pi(uv) \geq 0$

Exercise: Condition for using $\pi(u) = D(u, t)$ for a metric $D$?

Bidirectional A* : ALT [Goldberg, Harelsson ’05]

(ALT = A*, Landmarks, Triangle inequality)

- $\ell_\pi(uv) = \ell_\pi(vu) \iff \pi + \frac{\pi'}{2} = \text{cte}$
  
  (ex: $\pi' = (\pi - \frac{\pi}{2})/2$ and $\pi' = (\frac{\pi}{2} - \pi)/2$)

- or different stopping condition

- $\pi$ from landmarks (better than using coordinates):
  
  $\pi(u) = \max_{x \in X} d_G(u, x) - d_G(t, x)$

Exercise: Prove it works for bidir. A*.
A* [Hart, Nilsson, Raphael ’68]

Shortest path algorithm with prediction:

Potential function $\pi(u) \approx d(u, t)$.

Dijkstra ($G_\pi, s, t$) with $\ell_\pi(uv) = \ell(uv) - (\pi(u) - \pi(v))$.

$\pi(s, t) = d(s, t) - (\pi(s) - \pi(t))$.

Scan $u$ with $d(u) + \pi(u)$ min.

$\pi$ feasible if $\forall uv \in E(G), \ell_\pi(uv) \geq 0$

Exercise: Condition for using $\pi(u) = D(u, t)$ for a metric $D$?

Bidirectional $A^*$: ALT [Goldberg, Harelsson ’05]

(ALT = $A^*$, Landmarks, Triangle inequality)

- $\ell_\pi(uv) = \leftarrow \pi(vu) \iff \pi + \leftarrow \pi = cte$
  (ex: $\pi' = (\pi - \leftarrow \pi)/2$ and $\pi' = (\leftarrow \pi - \pi)/2$)
- or different stopping condition
- $\pi$ from landmarks (better than using coordinates):
  $\pi(u) = \max_{x \in X} d_G(u, x) - d_G(t, x)$

Exercise: Prove it works for bidir. $A^*$.
Reach pruning [Gutman ’04] revisited [Goldberg, Kaplan, Werneck ’05]

\[
\text{Reach}(u) = \max_{(s,t) | u \in P_{st}} \min \{ d(s, u), d(u, t) \}
\]

In bidir. Dijkstra, when scanning \( u \):

- **Prune** \( v \) s.t. \( \text{Reach}(v) < \min \{ d(u) + \ell(uv), \bar{r} \} \).

Add **shortcuts**:

- Tie break: fewer links is shorter.

**Exercise**: how to get shortest path from \( s \) to \( t \)?

Pre-compute **reach upper bounds**:

- Eliminate nodes with reach \( \leq \delta \).
- Shortcut paths with degree 2 nodes.
- Repeat with larger \( \delta \).
Reach pruning [Gutman ’04] revisit [Goldberg, Kaplan, Werneck ’05]

\[ \text{Reach}(u) = \max_{(s,t) \in Pst} \min \{ d(s, u), d(u, t) \} \]

In bidir. Dijkstra, when scanning \( u \):
- **Prune** \( v \) s.t. \( \text{Reach}(v) < \min \{ d(u) + \ell(uv), \overleftarrow{r} \} \).

Add **shortcuts**:
- Tie break: fewer links is shorter.

**Exercise**: how to get shortest path from \( s \) to \( t \)?

Pre-compute **reach upper bounds**:
- Eliminate nodes with reach \( \leq \delta \).
- Shortcut paths with degree 2 nodes.
- Repeat with larger \( \delta \).
Reach pruning [Gutman ‘04] revisited [Goldberg, Kaplan, Werneck ‘05]

\[ \text{Reach}(u) = \max_{(s,t) \mid u \in P_{st}} \min \{ d(s, u), d(u, t) \} \]

In bidir. Dijkstra, when scanning \( u \):
- **Prune** \( v \) s.t. \( \text{Reach}(v) < \min \{ d(u) + \ell(uv), r \} \).

Add **shortcuts**:
- Tie break: fewer links is shorter.

**Exercise**: how to get shortest path from \( s \) to \( t \)?

Pre-compute **reach upper bounds**:
- Eliminate nodes with reach \( \leq \delta \).
- Shortcut paths with degree 2 nodes.
- Repeat with larger \( \delta \).
Reach pruning [Gutman ’04] revisited [Goldberg, Kaplan, Werneck ’05]

Reach\( (u) = \max_{(s,t) | u \in P_{st}} \min \{ d(s, u), d(u, t) \} \)

In bidir. Dijkstra, when scanning \( u \):
- Prune \( v \) s.t. Reach\( (v) < \min \{ d(u) + \ell(uv), \overleftarrow{r} \} \).

Add shortcuts:
- Tie break: fewer links is shorter.

**Exercise**: how to get shortest path from \( s \) to \( t \)?

Pre-compute reach upper bounds:
- Eliminate nodes with reach \( \leq \delta \).
- Shortcut paths with degree 2 nodes.
- Repeat with larger \( \delta \).
Reach pruning [Gutman '04] revisited [Goldberg, Kaplan, Werneck '05]

\[ \text{Reach}(u) = \max_{(s,t) \mid u \in P_{st}} \min \{d(s,u), d(u,t)\} \]

In bidir. Dijkstra, when scanning \( u \):
- **Prune** \( v \) s.t. \( \text{Reach}(v) < \min \{d(u) + \ell(uv), r\} \).

Add **shortcuts**:
- Tie break: fewer links is shorter.

**Exercise**: how to get shortest path from \( s \) to \( t \)?

Pre-compute **reach upper bounds**:
- Eliminate nodes with reach \( \leq \delta \).
- Shortcut paths with degree 2 nodes.
- Repeat with larger \( \delta \).
Contraction Hierarchies [Geisberger, Sanders, Shultes, Delling ’05-08]

**Node ordering** \( \pi : u_1 < \cdots < u_n \)

Contract successively \( u_i \):
- add shortcut \( vw \) for \( v, w \in N(u_i) \) (if needed),
- remove \( u_i \) (distances are preserved in remaining graph).

**Query** : bidir. Dij. in \( G^{+\uparrow} \) and \( G^{+\uparrow} \).
- \( G^+ \): graph + shortcuts
- \( \uparrow \): follow \( uv \) if \( u <_\pi v \)

**Finding** \( \pi \):
- small degree + levels (MIS),
- min fill-in (greedy treewidth dec.),
- small balanced separators \( O(n \log n) \) shortcuts if planar.

**Exercise** : bound the number of shortcuts if any subgraph of \( G \) has an \( O(n^{\epsilon}) \) balanced separator and maximum degree \( \Delta \).

**Open pb** : link between small Reach and small CH?
Contraction Hierarchies [Geisberger, Sanders, Shultes, Delling ’05-08]

Node ordering $\pi : u_1 < \cdots < u_n$

Contract successively $u_i$:
- add shortcut $vw$ for $v, w \in N(u_i)$ (if needed),
- remove $u_i$ (distances are preserved in remaining graph).

Query: bidir. Dij. in $G^+\uparrow$ and $G^{+\uparrow}$.
- $G^+$: graph + shortcuts
- $\uparrow$: follow $uv$ if $u <_\pi v$

Finding $\pi$:
- small degree + levels (MIS),
- min fill-in (greedy treewidth dec.),
- small balanced separators ($O(n \log n)$ shortcuts if planar).

Exercise: bound the number of shortcuts if any subgraph of $G$ has an $O(n^\varepsilon)$ balanced separator and maximum degree $\Delta$.

Open pb: link between small Reach and small CH?
Contraction Hierarchies [Geisberger, Sanders, Shultes, Delling ’05-08]

Node ordering $\pi : u_1 < \cdots < u_n$

Contract successively $u_i$:
- add shortcut $vw$ for $v, w \in N(u_i)$ (if needed),
- remove $u_i$ (distances are preserved in remaining graph).

Query: bidir. Dij. in $G^+\uparrow$ and $G^+\downarrow$.
- $G^+$: graph + shortcuts
- $\uparrow$: follow $uv$ if $u \leq_\pi v$

Finding $\pi$:
- small degree + levels (MIS),
- min fill-in (greedy treewidth dec.),
- small balanced separators ($O(n \log n)$ shortcuts if planar).

Exercise: bound the number of shortcuts if any subgraph of $G$ has an $O(n^{\varepsilon})$ balanced separator and maximum degree $\Delta$.

Open pb: link between small Reach and small CH?
**Contraction Hierarchies** [Geisberger, Sanders, Shultes, Delling ’05-08]

Node ordering $\pi : u_1 < \cdots < u_n$

**Contract** successively $u_i :$
- add shortcut $vw$ for $v, w \in N(u_i)$ (if needed),
- remove $u_i$ (distances are preserved in remaining graph).

**Query** : bidir. Dij. in $G^+\uparrow$ and $G^{+\uparrow}$.
- $G^+$ : graph + shortcuts
- $\uparrow$ : follow $uv$ if $u \leq_\pi v$

**Finding** $\pi :$
- small degree + levels (MIS),
- min fill-in (greedy treewidth dec.),
- small balanced separators ($O(n \log n)$ shortcuts if planar).

**Exercise** : bound the number of shortcuts if any subgraph of $G$ has an $O(n^\varepsilon)$ balanced separator and maximum degree $\Delta$.

**Open pb** : link between small Reach and small CH?
Contraction Hierarchies [Geisberger, Sanders, Shultes, Delling ’05-08]

Node ordering $\pi : u_1 < \cdots < u_n$

Contract successively $u_i$:
- add shortcut $vw$ for $v, w \in N(u_i)$ (if needed),
- remove $u_i$ (distances are preserved in remaining graph).

Query: bidir. Dij. in $G^{+\uparrow}$ and $G^{\uparrow\uparrow}$.
- $G^+:$ graph + shortcuts
- $\uparrow: \text{follow } uv \text{ if } u \leq_{\pi} v$

Finding $\pi$:
- small degree + levels (MIS),
- min fill-in (greedy treewidth dec.),
- small balanced separators ($O(n \log n)$ shortcuts if planar).

Exercise: bound the number of shortcuts if any subgraph of $G$ has an $O(n^{\epsilon})$ balanced separator and maximum degree $\Delta$.

Open pb: link between small Reach and small CH?
Contraction Hierarchies [Geisberger, Sanders, Shultes, Delling ’05-08]

Node ordering $\pi : u_1 < \cdots < u_n$

Contract successively $u_i$:
- add shortcut $vw$ for $v, w \in N(u_i)$ (if needed),
- remove $u_i$ (distances are preserved in remaining graph).

Query: bidir. Dij. in $G^\uparrow$ and $G^\uparrow$.
- $G^\uparrow$: graph + shortcuts
- $\uparrow$: follow $uv$ if $u <_\pi v$

Finding $\pi$:
- small degree + levels (MIS),
- min fill-in (greedy treewidth dec.),
- small balanced separators ($O(n \log n)$ shortcuts if planar).

Exercise: bound the number of shortcuts if any subgraph of $G$ has an $O(n^\varepsilon)$ balanced separator and maximum degree $\Delta$.

Open pb: link between small Reach and small CH?
**CH complexity for planar graphs 1/3.**

**Theorem [Lipton, Tarjan ‘79]** Every planar graph \( G \) has a 2/3-balanced separator \( S_0 \) of size \( O(\sqrt{n}) \).

**Elimination ordering** \( \pi \) : recursively order each connected component and then add nodes in \( S_0 \) in any order. This results in a tree of separators of depth \( O(\log n) \). (Nested dissection as in [Gilbert, Tarjan ‘87].)

**Rq :** all shortcuts occur between a node and an ancestor (possibly the node itself).

**Corollary :** The nodes visited during a pruned Dijkstra from a node \( s \) at depth \( k \) are all in the separators in the branch of \( s \) and query time is \( O(\sum_{i=0}^{k} s_i) = O(\sqrt{n}) \) (where \( s_i = c\sqrt{(2/3)^i}n \) is the maximum size of a separator at depth \( i \)).
Theorem [Lipton, Tarjan ’79] Every planar graph $G$ has a 2/3-balanced separator $S_0$ of size $O(\sqrt{n})$.

Elimination ordering $\pi$: recursively order each connected component and then add nodes in $S_0$ in any order. This results in a tree of separators of depth $O(\log n)$. (Nested dissection as in [Gilbert, Tarjan ’87].)

Rq: all shortcuts occur between a node and an ancestor (possibly the node itself).

Corollary: The nodes visited during a pruned Dijkstra from a node $s$ at depth $k$ are all in the separators in the branch of $s$ and query time is $O(\sum_{i=0}^{k} s_i) = O(\sqrt{n})$ (where $s_i = c\sqrt{(2/3)^i}n$ is the maximum size of a separator at depth $i$).
Theorem [Lipton, Tarjan ’79] Every planar graph $G$ has a 2/3-balanced separator $S_0$ of size $O(\sqrt{n})$.

Elimination ordering $\pi$: recursively order each connected component and then add nodes in $S_0$ in any order. This results in a tree of separators of depth $O(\log n)$. (Nested dissection as in [Gilbert, Tarjan ’87].)

Rq: all shortcuts occur between a node and an ancestor (possibly the node itself).

Corollary: The nodes visited during a pruned Dijkstra from a node $s$ at depth $k$ are all in the separators in the branch of $s$ and query time is $O(\sum_{i=0}^{k} s_i) = O(\sqrt{n})$ (where $s_i = c(\sqrt{2/3})^i n$ is the maximum size of a separator at depth $i$).
CH complexity for planar graphs 1/3.

**Theorem** [Lipton, Tarjan ’79] Every planar graph $G$ has a 2/3-balanced separator $S_0$ of size $O(\sqrt{n})$.

**Elimination ordering** $\pi$: recursively order each connected component and then add nodes in $S_0$ in any order. This results in a tree of separators of depth $O(\log n)$. (Nested dissection as in [Gilbert, Tarjan ’87].)

**Rq:** all shortcuts occur between a node and an ancestor (possibly the node itself).

**Corollary:** The nodes visited during a pruned Dijkstra from a node $s$ at depth $k$ are all in the separators in the branch of $s$ and query time is $O(\sum_{i=0}^{k} s_i) = O(\sqrt{n})$ (where $s_i = c\sqrt{(2/3)^i}n$ is the maximum size of a separator at depth $i$).
Lemma: There are $O(n)$ shortcuts with an extremity in $S_0$.

Rq1: At most $|S_0|^2 = O(n)$ inside $S_0$.

Rq2: Each shortcut with a node $u$ at depth $k$ is the result of the contraction of nodes in the subtree rooted at the separator $S$ containing $u$.

Def: define the bipartite graph $G_k$ with vertex set $S_0 \cup D_k$ where $D_k$ is the set of separators at depth $k$ and an edge $(v, S)$ if there is a node $v \in S_0$ and a separator $S \in D_k$ such that there exists a shortcut from $u \in S$ to $v$.

Rq3: $G_k$ is planar because it can be obtained from $G$ by removing edges inside $S_0$, removing nodes at depth within 1 and $k - 1$ and contracting edges between nodes at depth $k$ or deeper. We thus have $m(G_k) \leq 3n(G_k) - 6$ (Euler & $3f \leq 2m$).

Rq4: The number of shortcuts with an extremity at depth $k$ is $N_k \leq \sum_{S \in D_k} |S| \deg_{G_k}(S)$. If $D'_k = \{S : \deg_{G_k}(S) > 3\}$, then $N_k \leq \sum_{S} 3|S| + \sum_{S \in D'_k} s_k(\deg_{G_k}(S) - 3) \leq 3n_k + s_k(m(G'_k) - 3|D'_k|)$ where $G'_k = G_k[D'_k \cup S_0]$ satisfies $m(G'_k) \leq 3n(G'_k)$ implying $N_k \leq 3n_k + 3s_k|S_0|$. The Lemma follows from $\sum_k s_k = O(\sqrt{n})$. 
**Lemma:** There are $O(n)$ shortcuts with an extremity in $S_0$.

**Rq1:** At most $|S_0|^2 = O(n)$ inside $S_0$.

**Rq2:** Each shortcut with a node $u$ at depth $k$ is the result of the contraction of nodes in the subtree rooted at the separator $S$ containing $u$.

**Def:** define the bipartite graph $G_k$ with vertex set $S_0 \cup D_k$ where $D_k$ is the set of separators at depth $k$ and an edge $(v, S)$ if there is a node $v \in S_0$ and a separator $S \in D_k$ such that there exists a shortcut from $u \in S$ to $v$.

**Rq3:** $G_k$ is planar because it can be obtained from $G$ by removing edges inside $S_0$, removing nodes at depth within 1 and $k-1$ and contracting edges between nodes at depth $k$ or deeper. We thus have $m(G_k) \leq 3n(G_k) - 6$ (Euler & $3f \leq 2m$).

**Rq4:** The number of shortcuts with an extremity at depth $k$ is $N_k \leq \sum_{S \in D_k} |S| \deg_{G_k}(S)$. If $D'_k = \{S : \deg_{G_k}(S) > 3\}$, then $N_k \leq \sum S 3|S| + \sum_{S \in D_k} s_k(\deg_{G_k}(S) - 3) \leq 3n_k + s_k(m(G'_k) - 3|D'_k|)$ where $G'_k = G_k[D'_k \cup S_0]$ satisfies $m(G'_k) \leq 3n(G'_k)$ implying $N_k \leq 3n_k + 3s_k|S_0|$. The Lemma follows from $\sum_k s_k = O(\sqrt{n})$. 

$\leftarrow ? \Rightarrow$
Lemma: There are $O(n)$ shortcuts with an extremity in $S_0$.

Rq1: At most $|S_0|^2 = O(n)$ inside $S_0$.

Rq2: Each shortcut with a node $u$ at depth $k$ is the result of the contraction of nodes in the subtree rooted at the separator $S$ containing $u$.

Def: define the bipartite graph $G_k$ with vertex set $S_0 \cup D_k$ where $D_k$ is the set of separators at depth $k$ and an edge $(v, S)$ if there is a node $v \in S_0$ and a separator $S \in D_k$ such that there exists a shortcut from $u \in S$ to $v$.

Rq3: $G_k$ is planar because it can be obtained from $G$ by removing edges inside $S_0$, removing nodes at depth within 1 and $k - 1$ and contracting edges between nodes at depth $k$ or deeper. We thus have $m(G_k) \leq 3n(G_k) - 6$ (Euler $&$ $3f \leq 2m$).

Rq4: The number of shortcuts with an extremity at depth $k$ is $N_k \leq \sum_{S \in D_k} |S| \deg_{G_k}(S)$. If $D'_k = \{S : \deg_{G_k}(S) > 3\}$, then

$$N_k \leq \sum_S 3|S| + \sum_{S \in D'_k} s_k(\deg_{G_k}(S) - 3) \leq 3n_k + s_k(m(G'_k) - 3|D'_k|)$$

where $G'_k = G_k[D'_k \cup S_0]$ satisfies $m(G'_k) \leq 3n(G'_k)$ implying $N_k \leq 3n_k + 3s_k|S_0|$. The Lemma follows from $\sum k s_k = O(\sqrt{n})$. 
Lemma: There are $O(n)$ shortcuts with an extremity in $S_0$.

Rq1: At most $|S_0|^2 = O(n)$ inside $S_0$.

Rq2: Each shortcut with a node $u$ at depth $k$ is the result of the contraction of nodes in the subtree rooted at the separator $S$ containing $u$.

Def: define the bipartite graph $G_k$ with vertex set $S_0 \cup D_k$ where $D_k$ is the set of separators at depth $k$ and an edge $(v, S)$ if there is a node $v \in S_0$ and a separator $S \in D_k$ such that there exists a shortcut from $u \in S$ to $v$.

Rq3: $G_k$ is planar because it can be obtained from $G$ by removing edges inside $S_0$, removing nodes at depth within 1 and $k-1$ and contracting edges between nodes at depth $k$ or deeper. We thus have $m(G_k) \leq 3n(G_k) - 6$ (Euler & $3f \leq 2m$).

Rq4: The number of shortcuts with an extremity at depth $k$ is $N_k \leq \sum_{S \in D_k} |S| \deg_{G_k}(S)$. If $D'_k = \{S : \deg_{G_k}(S) > 3\}$, then $N_k \leq \sum_S 3|S| + \sum_{S \in D_k} s_k(\deg_{G_k}(S) - 3) \leq 3n_k + s_k(m(G'_k) - 3|D'_k|)$ where $G'_k = G_k[D'_k \cup S_0]$ satisfies $m(G'_k) \leq 3n(G'_k)$ implying $N_k \leq 3n_k + 3s_k|S_0|$. The Lemma follows from $\sum_k s_k = O(\sqrt{n})$. 

$\leftarrow ? \Rightarrow$
Lemma: There are $O(n)$ shortcuts with an extremity in $S_0$.

Rq1: At most $|S_0|^2 = O(n)$ inside $S_0$.

Rq2: Each shortcut with a node $u$ at depth $k$ is the result of the contraction of nodes in the subtree rooted at the separator $S$ containing $u$.

Def: define the bipartite graph $G_k$ with vertex set $S_0 \cup D_k$ where $D_k$ is the set of separators at depth $k$ and an edge $(v, S)$ if there is a node $v \in S_0$ and a separator $S \in D_k$ such that there exists a shortcut from $u \in S$ to $v$.

Rq3: $G_k$ is planar because it can be obtained from $G$ by removing edges inside $S_0$, removing nodes at depth within 1 and $k - 1$ and contracting edges between nodes at depth $k$ or deeper. We thus have $m(G_k) \leq 3n(G_k) - 6$ (Euler & $3f \leq 2m$).

Rq4: The number of shortcuts with an extremity at depth $k$ is $N_k \leq \sum_{S \in D_k} |S|\deg_{G_k}(S)$. If $D_k' = \{S : \deg_{G_k}(S) > 3\}$, then $N_k \leq \sum_{S} 3|S| + \sum_{S \in D_k'} s_k(\deg_{G_k}(S) - 3) \leq 3n_k + s_k(m(G'_k) - 3|D_k'|)$ where $G'_k = G_k[D_k' \cup S_0]$ satisfies $m(G'_k) \leq 3n(G'_k)$ implying $N_k \leq 3n_k + 3s_k|S_0|$. The Lemma follows from $\sum_k s_k = O(\sqrt{n})$. 

\[ \sum_{k} s_k = O(\sqrt{n}) \]
Lemma: There are $O(n)$ shortcuts with an extremity in $S_0$.

Rq1: At most $|S_0|^2 = O(n)$ inside $S_0$.

Rq2: Each shortcut with a node $u$ at depth $k$ is the result of the contraction of nodes in the subtree rooted at the separator $S$ containing $u$.

Def: define the bipartite graph $G_k$ with vertex set $S_0 \cup D_k$ where $D_k$ is the set of separators at depth $k$ and an edge $(v, S)$ if there is a node $v \in S_0$ and a separator $S \in D_k$ such that there exists a shortcut from $u \in S$ to $v$.

Rq3: $G_k$ is planar because it can be obtained from $G$ by removing edges inside $S_0$, removing nodes at depth within 1 and $k-1$ and contracting edges between nodes at depth $k$ or deeper. We thus have $m(G_k) \leq 3n(G_k) - 6$ (Euler & $3f \leq 2m$).

Rq4: The number of shortcuts with an extremity at depth $k$ is $N_k \leq \sum_{S \in D_k} |S| \deg_{G_k}(S)$. If $D'_k = \{S : \deg_{G_k}(S) > 3\}$, then $N_k \leq \sum_S 3|S| + \sum_{S \in D'_k} s_k(\deg_{G_k}(S) - 3) \leq 3n_k + s_k(m(G'_k) - 3|D'_k|)$ where $G'_k = G_k[D'_k \cup S_0]$ satisfies $m(G'_k) \leq 3n(G'_k)$ implying $N_k \leq 3n_k + 3s_k|S_0|$. The Lemma follows from $\sum_k s_k = O(\sqrt{n})$. 
Corollary: The number of shortcuts generated by contracting $G$ according to $\pi$ is $O(n \log n)$.

Rq1: Similarly to the previous Lemma, for each separator $S$ root of a subtree of $n'$ nodes, there are $O(n')$ shortcuts with an extremity in $S$.

Rq2: When summing all subtree sizes, a node is counted $O(\log n)$ times.
CH complexity for planar graphs 3/3.

**Corollary**: The number of shortcuts generated by contracting $G$ according to $\pi$ is $O(n \log n)$.

**Rq1**: Similarly to the previous Lemma, for each separator $S$ root of a subtree of $n'$ nodes, there are $O(n')$ shortcuts with an extremity in $S$.

**Rq2**: When summing all subtree sizes, a node is counted $O(\log n)$ times.
Corollary: The number of shortcuts generated by contracting $G$ according to $\pi$ is $O(n \log n)$.

Rq1: Similarly to the previous Lemma, for each separator $S$ root of a subtree of $n'$ nodes, there are $O(n')$ shortcuts with an extremity in $S$.

Rq2: When summing all subtree sizes, a node is counted $O(\log n)$ times.
Transit node routing [Bast, Funke, Matjevic '07; Sanders, Schultex '09]

Transit nodes $T$ so that any long distance path goes through a transit node $x \in T$. Pre-compute all distances $d(x, y)$ for $x, y \in T$.

Access nodes $A(v) \subseteq T$: any long path from/to $v$ goes through $x \in A(v)$.

Long distance query $(s, t)$:
\[
\min_{x \in A(s), y \in A(t)} d(s, x) + d(x, y) + d(y, t)
\]

Local query $(s, t)$: Use bidirectional Dijkstra or even CH.
Transit node routing [Bast, Funke, Matjevic ’07; Sanders, Schultex ’09]

Transit nodes $T$ so that any long distance path goes through a transit node $x \in T$. Pre-compute all distances $d(x, y)$ for $x, y \in T$.

Access nodes $A(v) \subseteq T$: any long path from/to $v$ goes through $x \in A(v)$.

Long distance query $(s, t)$:
$\min_{x \in A(s), y \in A(t)} d(s, x) + d(x, y) + d(y, t)$

Local query $(s, t)$: Use bidirectional Dijkstra or even CH.
Transit node routing [Bast, Funke, Matjievic ’07; Sanders, Schultex ’09]

Transit nodes $T$ so that any long distance path goes through a transit node $x \in T$. Pre-compute all distances $d(x, y)$ for $x, y \in T$.

Access nodes $A(v) \subseteq T$: any long path from/to $v$ goes through $x \in A(v)$.

Long distance query $(s, t)$:
$$\min_{x \in A(s), y \in A(t)} d(s, x) + d(x, y) + d(y, t)$$

Local query $(s, t)$: Use bidirectional Dijkstra or even CH.
Transit node routing [Bast, Funke, Matjevic ‘07; Sanders, Schultex ‘09]

Transit nodes $T$ so that any long distance path goes through a transit node $x \in T$. Pre-compute all distances $d(x, y)$ for $x, y \in T$.

Access nodes $A(v) \subseteq T$: any long path from/to $v$ goes through $x \in A(v)$.

Long distance query $(s, t)$:
$\min_{x \in A(s), y \in A(t)} d(s, x) + d(x, y) + d(y, t)$

Local query $(s, t)$: Use bidirectional Dijkstra or even CH.
Highway dimension [Abraham, Delling, Fiat, Goldberg, Werneck ’10-13]

**Graph property** ensuring efficient ordering for CH and fast pruned bidir. Dijkstra.

**Definition**

Highway dimension \( h = \max_{u,r} \min_H \text{ hitting set of } \mathcal{P}_{ur} |H| \)

where \( \mathcal{P}_{ur} = \{ P \in \mathcal{P}_r \mid \overline{P} \cap B(u, 2r) \neq \emptyset \} \), \( \mathcal{P}_r = \{ P \mid \ell(P) > r \} \),

and \( \overline{P} \) is any shortest path extending \( P \) by 0 or 1 edge at each extremity.

**Theorem**

Any graph \( G \) with highway dimension \( h \) and diameter \( D \)

admits a node ordering \( \pi \) s.t. \( CH_\pi \) produces at most \( O(nh \log D) \) shortcuts and \( CH_\pi + \text{RP} \) bidir. Dij. visits \( O(h \log D) \) nodes. (RP : prune \( v \) s.t. \( d(v) > 2^{\text{level}(v)} \))

**Lemma**

For all \( r \), \( G \) has an \((h, r)\)-sparse hitting set, i.e. a set \( C \) s.t.

\( C \cap P \neq \emptyset \) for all \( P \in \mathcal{P}_r \) and \( |B(u, 2r) \cap C| \leq h \) for all \( u \in V(G) \).
Highway dimension [Abraham, Delling, Fiat, Goldberg, Werneck '10-13]

**Graph property** ensuring efficient ordering for CH and fast pruned bidir. Dijkstra.

**Definition**

**Highway dimension** \( h = \max_{u,r} \min_H \text{hitting set of } P_{ur} | H | \)

where \( P_{ur} = \{ P \in P_r | \bar{P} \cap B(u, 2r) \neq \emptyset \} \), \( P_r = \{ P | \ell(P) > r \} \),

and \( \bar{P} \) is any shortest path extending \( P \) by 0 or 1 edge at each extremity.

**Theorem**

Any graph \( G \) with highway dimension \( h \) and diameter \( D \) admits a node ordering \( \pi \) s.t. \( CH_\pi \) produces at most \( O(nh \log D) \) shortcuts and \( CH_\pi + RP \) bidir. Dij. visits \( O(h \log D) \) nodes. (\( RP \) : prune \( v \) s.t. \( d(v) > 2^{\text{level}(v)} \))

**Lemma**

For all \( r \), \( G \) has an \( (h, r) \)-sparse hitting set, i.e. a set \( C \) s.t.

\( C \cap P \neq \emptyset \) for all \( P \in P_r \) and \( |B(u, 2r) \cap C| \leq h \) for all \( u \in V(G) \).
Highway dimension [Abraham, Delling, Fiat, Goldberg, Werneck ’10-13]

**Graph property** ensuring efficient ordering for CH and fast pruned bidir. Dijkstra.

**Definition**

Highway dimension $h = \max_{u,r} \min_{|H|} \text{hitting set of } P_{ur}$

where $P_{ur} = \{P \in P_r | \bar{P} \cap B(u, 2r) \neq \emptyset\}$, $P_r = \{P | \ell(P) > r\}$, and $\bar{P}$ is any shortest path extending $P$ by 0 or 1 edge at each extremity.

**Theorem**

Any graph $G$ with highway dimension $h$ and diameter $D$ admits a node ordering $\pi$ s.t. $CH_\pi$ produces at most $O(nh \log D)$ shortcuts and $CH_\pi + RP$ bidir. Dij. visits $O(h \log D)$ nodes. ($RP : \text{prune } v \text{ s.t. } d(v) > 2^{\text{level}(v)}$)

**Lemma**

For all $r$, $G$ has an $(h, r)$-sparse hitting set, i.e. a set $C$ s.t. $C \cap P \neq \emptyset$ for all $P \in P_r$ and $|B(u, 2r) \cap C| \leq h$ for all $u \in V(G)$. 

\[ \Rightarrow \]
Highway dimension [Abraham, Delling, Fiat, Goldberg, Werneck ’10-13]

**Graph property** ensuring efficient ordering for CH and fast pruned bidir. Dijkstra.

**Definition**
Highway dimension \( h = \max_{u,r} \min_{H} \text{hitting set of } \mathcal{P}_{ur} |H| \)
where \( \mathcal{P}_{ur} = \{ P \in \mathcal{P}_r | \bar{P} \cap B(u, 2r) \neq \emptyset \} \), \( \mathcal{P}_r = \{ P | \ell(\bar{P}) > r \} \),
and \( \bar{P} \) is any shortest path extending \( P \) by 0 or 1 edge at each extremity.

**Theorem**
Any graph \( G \) with highway dimension \( h \) and diameter \( D \) admits a node ordering \( \pi \) s.t. \( CH_{\pi} \) produces at most \( O(nh \log D) \) shortcuts and \( CH_{\pi} + RP \) bidir. Dij. visits \( O(h \log D) \) nodes. (\( RP : \text{prune } v \text{ s.t. } d(v) > 2^{\text{level}(v)} \))

**Lemma**
For all \( r \), \( G \) has an \((h, r)\)-sparse hitting set, i.e. a set \( C \) s.t. \( C \cap P \neq \emptyset \) for all \( P \in \mathcal{P}_r \) and \( |B(u, 2r) \cap C| \leq h \) for all \( u \in V(G) \).
Proof Lemma: Define $C$ as a minimum hitting set for $P_r$.

Idea Theorem: Construct a $(h, 2^{i-1})$-sparse hitting set $C_i$ for $i = 0, \ldots, \lceil \log D \rceil$.
Define $C_i' = C_i \setminus \cup_{j > i} C_j$ and $\pi = C_0', \ldots, C_{\lceil \log D \rceil}'$.

Rk1: For each shortcut $vw$ with $v \in C_i'$ and $w \in C_j'$ with $i \leq j$, we have $d(v, w) \leq 2^i$.

Cor1: Each node $v$ has at most $h \lceil \log D \rceil$ shortcuts.

Rk2: In CH bidir. Dijkstra query, prune $v \in C_i'$ s.t. $d(v) > 2^i$.

Cor2: At most $h$ nodes of $C_i'$ are visited.
**Proof Lemma:** Define $C$ as a minimum hitting set for $\mathcal{P}_r$. 

**Idea Theorem:** Construct a $(h, 2^{i-1})$-sparse hitting set $C_i$ for $i = 0, \ldots, \lceil \log D \rceil$.

Define $C'_i = C_i \setminus \bigcup_{j > i} C_j$ and $\pi = C'_0, \ldots, C'_{\lceil \log D \rceil}$.

**Rk1:** For each shortcut $vw$ with $v \in C'_i$ and $w \in C'_j$ with $i \leq j$, we have $d(v, w) \leq 2^i$.

**Cor1:** Each node $v$ has at most $h \lceil \log D \rceil$ shortcuts.

**Rk2:** In CH bidir. Dijkstra query, prune $v \in C'_i$ s.t. $d(v) > 2^i$.

**Cor2:** At most $h$ nodes of $C'_i$ are visited.
Proof Lemma: Define $C$ as a minimum hitting set for $\mathcal{P}_r$.

Idea Theorem: Construct a $(h, 2^{i-1})$-sparse hitting set $C_i$ for $i = 0, \ldots, \lceil \log D \rceil$.

Define $C' = C_i \setminus \bigcup_{j > i} C_j$ and $\pi = C'_0, \ldots, C'_{\lceil \log D \rceil}$.

Rk1: For each shortcut $vw$ with $v \in C'_i$ and $w \in C'_j$ with $i \leq j$, we have $d(v, w) \leq 2^i$.

Cor1: Each node $v$ has at most $h \lceil \log D \rceil$ shortcuts.

Rk2: In CH bidir. Dijkstra query, prune $v \in C'_i$ s.t. $d(v) > 2^i$.

Cor2: At most $h$ nodes of $C'_i$ are visited.
Proof Lemma: Define $C$ as a minimum hitting set for $P_r$.

Idea Theorem: Construct a $(h, 2^{i-1})$-sparse hitting set $C_i$ for $i = 0, \ldots, \lceil \log D \rceil$.

Define $C_i' = C_i \setminus \bigcup_{j > i} C_j$ and $\pi = C'_0, \ldots, C'_{\lceil \log D \rceil}$.

Rk1: For each shortcut $vw$ with $v \in C'_i$ and $w \in C'_j$ with $i \leq j$, we have $d(v, w) \leq 2^i$.

Cor1: Each node $v$ has at most $h \lceil \log D \rceil$ shortcuts.

Rk2: In CH bidir. Dijkstra query, prune $v \in C'_i$ s.t. $d(v) > 2^i$.

Cor2: At most $h$ nodes of $C'_i$ are visited.
Proof Lemma: Define $C$ as a minimum hitting set for $P_r$.

Idea Theorem: Construct a $(h, 2^{i-1})$-sparse hitting set $C_i$ for $i = 0, \ldots, \lceil \log D \rceil$.
Define $C'_i = C_i \setminus \bigcup_{j > i} C_j$ and $\pi = C'_0, \ldots, C'_{\lceil \log D \rceil}$.

Rk1: For each shortcut $vw$ with $v \in C'_i$ and $w \in C'_j$ with $i \leq j$, we have $d(v, w) \leq 2^i$.

Cor1: Each node $v$ has at most $h \lceil \log D \rceil$ shortcuts.

Rk2: In CH bidir. Dijkstra query, prune $v \in C'_i$ s.t. $d(v) > 2^i$.

Cor2: At most $h$ nodes of $C'_i$ are visited.
Proof Lemma: Define $C$ as a minimum hitting set for $P_r$.

Idea Theorem: Construct a $(h, 2^{i-1})$-sparse hitting set $C_i$ for $i = 0, \ldots, \lceil \log D \rceil$.
Define $C'_i = C_i \setminus \bigcup_{j > i} C_j$ and $\pi = C'_0, \ldots, C'_{\lceil \log D \rceil}$.

Rk1: For each shortcut $vw$ with $v \in C'_i$ and $w \in C'_j$ with $i \leq j$, we have $d(v, w) \leq 2^i$.

Cor1: Each node $v$ has at most $h \lceil \log D \rceil$ shortcuts.

Rk2: In CH bidir. Dijkstra query, prune $v \in C'_i$ s.t. $d(v) > 2^i$.

Cor2: At most $h$ nodes of $C'_i$ are visited.
A striking remark on pruned Dijkstra

Rk : The $O(h \log D)$ bound on pruned Dijkstra from $s$ holds even without the bidir. stopping condition.

Idea : Pre-compute this pruned Dijkstra $D_v$ for all $v$!

Result : For any $s, t$ the distance (and the shortest path) from $s$ to $t$ can be computed from $D_s$ and $D_t$.

The collection $(D_v)_{v \in V}$ form a distance labeling of $G$ (subject of next lecture).
A striking remark on pruned Dijkstra

**Remark (R_k):** The $O(h \log D)$ bound on pruned Dijkstra from $s$ holds even without the bidirectional stopping condition.

**Idea:** Pre-compute this pruned Dijkstra $D_v$ for all $v$!

**Result:** For any $s, t$ the distance (and the shortest path) from $s$ to $t$ can be computed from $D_s$ and $D_t$.

The collection $(D_v)_{v \in V}$ form a distance labeling of $G$ (subject of next lecture).
A striking remark on pruned Dijkstra

**Rk**: The $O(h \log D)$ bound on pruned Dijkstra from $s$ holds even without the bidir. stopping condition.

**Idea**: Pre-compute this pruned Dijkstra $D_v$ for all $v$!

**Result**: For any $s, t$ the distance (and the shortest path) from $s$ to $t$ can be computed from $D_s$ and $D_t$.

The collection $(D_v)_{v \in V}$ form a distance labeling of $G$ (subject of next lecture).
A striking remark on pruned Dijkstra

**Rk**: The $O(h \log D)$ bound on pruned Dijkstra from $s$ holds even without the bidirectional stopping condition.

**Idea**: Pre-compute this pruned Dijkstra $D_v$ for all $v$!

**Result**: For any $s, t$ the distance (and the shortest path) from $s$ to $t$ can be computed from $D_s$ and $D_t$.

The collection $(D_v)_{v \in V}$ form a distance labeling of $G$ (subject of next lecture).
Further reading

[Schild, Sommer 2015]  
On balanced separators in road networks.  
SEA 2015.  

[Delling, Goldberg, Pajor, Werneck 2017]  
Customizable route planning in road networks.  
Transportation Science 2017.  
Further reading

[Schild, Sommer 2015]
On balanced separators in road networks.
SEA 2015.

[Delling, Goldberg, Pajor, Werneck 2017]
Customizable route planning in road networks.
Transportation Science 2017.
Exercise for next week

**Contraction order matters**: Propose a bounded-degree tree graph of \( n \) nodes and a contraction order that produce contraction hierarchies of size \( \Theta(n^2) \). (Nodes are contracted one after another in order. When contracting a node \( v \) of the current graph \( G \), we add shortcuts \( uw \) to \( G \) for neighbors \( u, w \) of \( u \) such that

\[
d_G(u, w) = d_G(u, v) + d_G(v, w)
\]

and remove \( v \). The size of the contraction hierarchies is counted as the number of shortcuts added.)

Send a short argumented answer to laurent.viennot@inria.fr.

Answers to exercises will count in the final mark.
Exercise for next week

**Contraction order matters**: Propose a bounded-degree tree graph of $n$ nodes and a contraction order that produce contraction hierarchies of size $\Theta(n^2)$. (Nodes are contracted one after another in order. When contracting a node $v$ of the current graph $G$, we add shortcuts $uw$ to $G$ for neighbors $u, w$ of $u$ such that $d_G(u, w) = d_G(u, v) + d_G(v, w)$ and remove $v$. The size of the contraction hierarchies is counted as the number of shortcuts added.)

Send a short argumented answer to laurent.viennot@inria.fr.

Answers to exercises will count in the final mark.
Exercise for next week

Contraction order matters: Propose a bounded-degree tree graph of n nodes and a contraction order that produce contraction hierarchies of size $\Theta(n^2)$. (Nodes are contracted one after another in order. When contracting a node $v$ of the current graph $G$, we add shortcuts $uw$ to $G$ for neighbors $u, w$ of $u$ such that $d_G(u, w) = d_G(u, v) + d_G(v, w)$ and remove $v$. The size of the contraction hierarchies is counted as the number of shortcuts added.)

Send a short argumented answer to laurent.viennot@inria.fr.

Answers to exercises will count in the final mark.