Efficient Route Planning

SS 2012

Lecture 7, Wednesday June 13th, 2012

(Contraction Hierarchies, Part 2 of 2)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

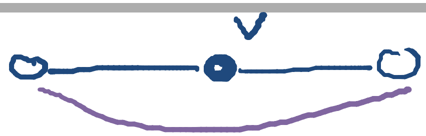
Overview of this lecture

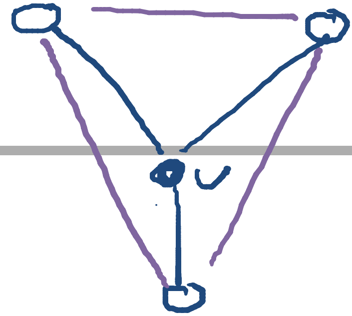
- Organizational
  - Feedback and results from Exercise Sheet 6 (CH, part 1)
- Contraction Hierarchies, Part 2 of 2
  - Query algorithm + example again
  - Correctness proof
  - Good node orderings
  - Exercise Sheet 7: implement a basic version of CH
    - Query algorithm (easy)
    - Simple node ordering (not hard either)
    - Use it to run 1000 queries and report results on the Wiki
    - Again: not much code, but you have to understand what you are doing

Your Feedback on Ex. Sheet 6 (CH, part 1)

- Summary / excerpts last checked June 13, 14:59
  - Was quite doable for most, difficultywise and timewise
  - Not much code, but many opportunities for mistakes
  - Graphic example in the lecture was helpful
  - Unit test for contractNode was most of the work
  - Thanks to the tutor for the great comments + answers

Your node contraction results





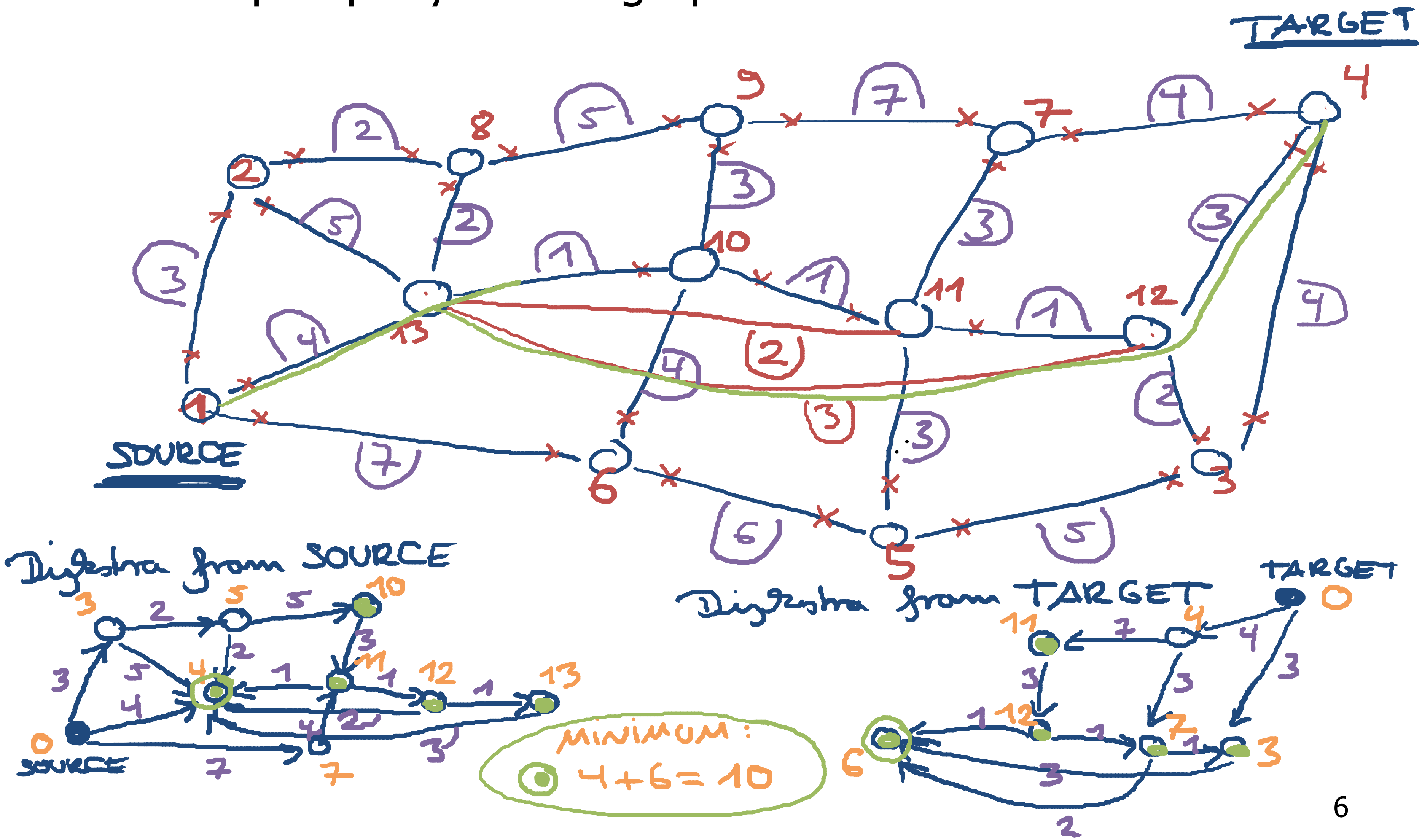
- Summary
  - Best contraction times indeed just a few μs per node
    - Note: 10μs / node → 10s / 1M nodes → 24s for BaWü
  - Number of shortcuts for 1000 random nodes
    - ~ 800 require 1, > 100 require 3, ~ 70 require 0
    - Note: 3 is much more frequent than 2 ... why?
  - Edge differences (ED) for these 1000 random nodes
    - Only ~ 10 have an ED of -2 (which is good)
    - Most have an ED of -1 or 0 or 1
  - These results suggests that picking nodes in a random order would add many more shortcuts than optimally possible

CH — Query algorithm 1/2 (from last lecture)

- Given $G^* = (V, E^*)$ and a source s and a target t
  - Define the upwards graph $G^*\uparrow = (V, \{(u, v) \in E^* : v > u\})$
  - Define the downwards graph $G^*\downarrow = (V, \{(u, v) \in E^* : v < u\})$
  - Do a full Dijkstra computation from s **forwards** in $G^*\uparrow$
  - Do a full Dijkstra computation from t **backwards** in $G^*\downarrow$
  - Let I be the set of nodes settled in **both** Dijkstras
  - Take $dist(s, t) = \min \{dist(s, v) + dist(v, t) : v \in I\}$
  - Is this correct and if yes why? ... slides 8 – 13
  - In the implementation, we need not construct $G^*\uparrow$ and $G^*\downarrow$ explicitly, we can just work on G^* ... slides 17 + 18
  - In symm. graphs backw. on $G^*\downarrow$ = forw. on $G^*\uparrow$ ... slide 7

CH — Query algorithm 2/2 (from last lecture)

- Example query on the graph from last lecture



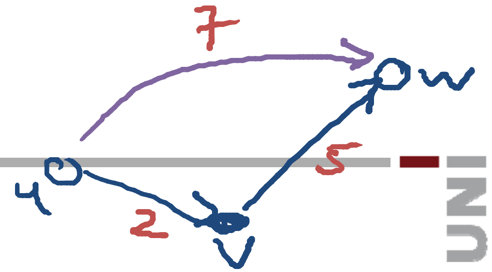
Symmetric graphs

- For symmetric graphs we only need $G^*\uparrow$
  - Recall the definitions:
    - Upwards graph $G^*\uparrow = (V, \{(u, v) \in E^* : v > u\})$
    - Downwards graph $G^*\downarrow = (V, \{(u, v) \in E^* : v < u\})$
  - A backwards search on an arbitary graph G is equivalent to a forward search on G with all arcs reversed
  - For symmetric graphs, G with all arcs reversed is = G
  - $G^*\downarrow$ with all arcs reversed is exactly $G^*\uparrow$
  - Hence a backwards search on $G^*\downarrow$ is exactly the same as a forward search on $G^*\uparrow$

CH — Correctness Proof 1/6

- First, the terminology from last lecture again
  - Let $u_1, ..., u_n$ be an **arbitrary** order of the nodes
    - we will see that the proof works for any order
  - Let $G = G_0$ be the initial graph
  - Let G_i be the graph obtained from G_{i-1} by contracting u_i that is, **without** u_i and adjacent arcs and **with** shortcuts
    - in particular therefore, G_i has $n - i$ nodes
  - In the end, let G^* be the original graph with **all nodes and arcs** and **all shortcuts** from any of the $G_1, G_2, ...$

CH — Correctness Proof 2/6



- Contraction preserves shortest paths
  - **Lemma 1:** For all $i = 1, ..., n$ we have for all $s, t \in G_i$ $dist_{G_i}(s, t) = dist_{G_{i-1}}(s, t)$
  - Corollary: hence by induction also $dist_{G_i}(s, t) = dist_G(s, t)$
- Proof of Lemma 1 ... it's pretty straightforward
  - Consider a SP from s to t in G_i
  - If this SP contains **no** shortcut that was added when u_i was contracted, we have the same path also in G_{i-1}
  - If it does contains a shortcut u, w added then, it means we have the path u, v, w in G_{i-1} with the same cost
  - This proves $dist_{G_{i-1}}(s, t) \leq dist_{G_i}(s, t)$
  - An analogous arguments proves $dist_{G_i}(s, t) \leq dist_{G_{i-1}}(s, t)$

CH — Correctness Proof 3/6

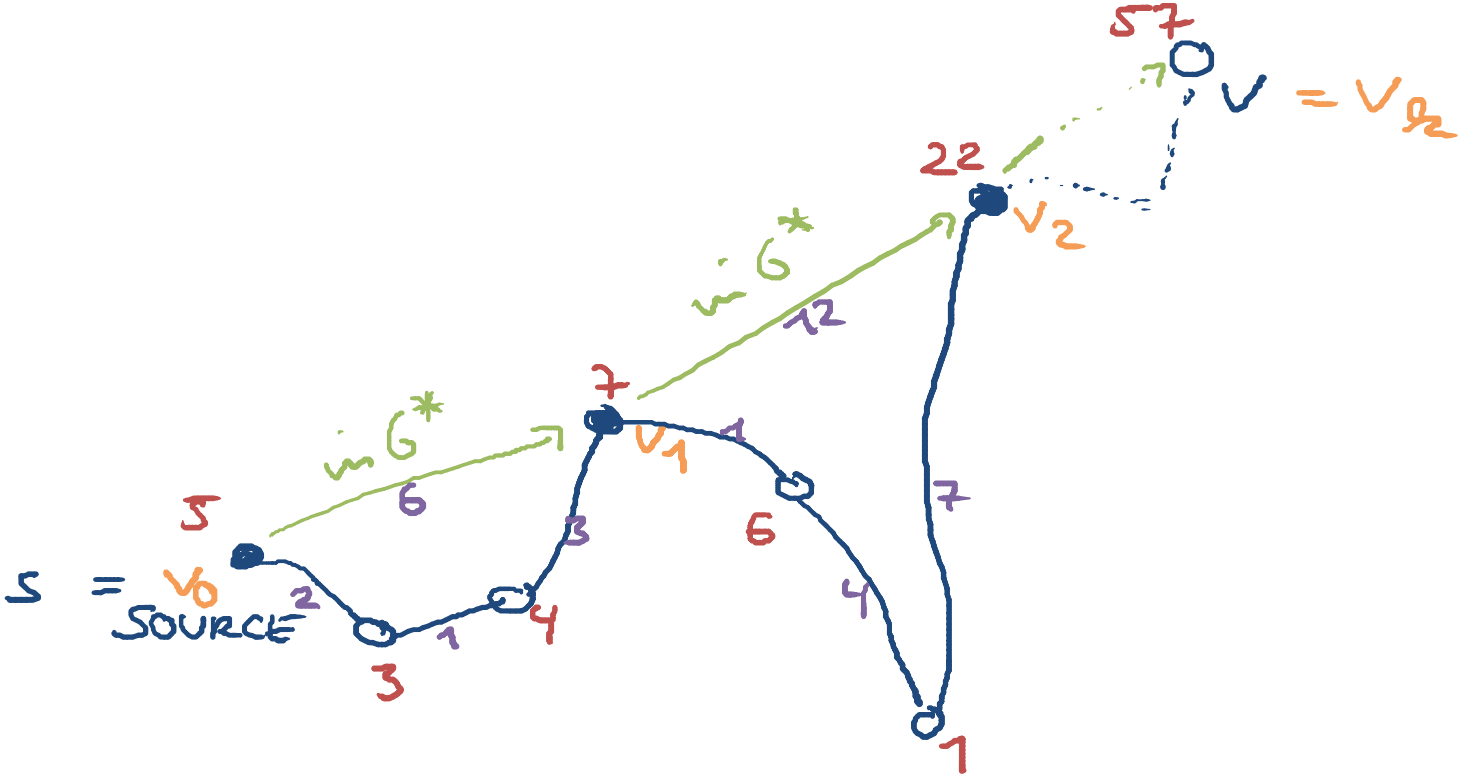
- Correctness of the query algorithm
  - **Lemma 2:** dist(s, t) = min {$dist_s[v] + dist_t[v]$: v ∈ I}
    where I is the set of nodes settled in both Dijkstra
- Proof of Lemma 2
  - Let **v** be the largest node (wrt the node ordering) on the SP from s to t in the original graph G
  - Consider the **prefix maxima** on the path from s to v, that is, the nodes $v_0 < v_1 < \ldots < v_k$ such that the SP is

    $s = v_0 \rightarrow^* v_1 \rightarrow^* v_2 \rightarrow^* \ldots \rightarrow^* v_k = v$

    where the subpaths $v_{i-1} \rightarrow^* v_i$ use only nodes $< v_{i-1}$

- Proof of Lemma 2, example of prefix maxima
  - From last slide: $s = v_0 \rightarrow^* v_1 \rightarrow^* v_2 \rightarrow^* \ldots \rightarrow^* v_k = v$

    where $v_{i-1} < v_i$ and $v_{i-1} \rightarrow^* v_i$ uses only nodes $< v_{i-1}$



CH — Correctness Proof 5/6

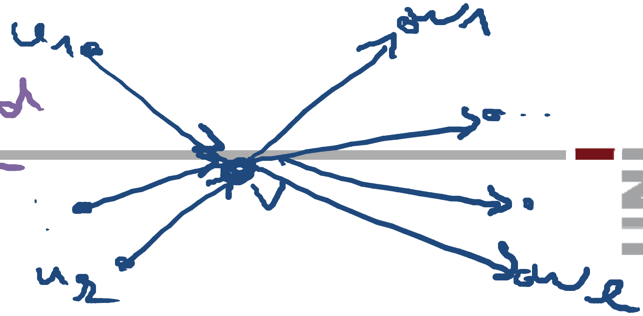
- Proof of Lemma 2 (continued)
  - From last slide: $s = v_0 \rightarrow^* v_1 \rightarrow^* v_2 \rightarrow^* \ldots \rightarrow^* v_k = v$
  - We prove that for each $i = 1, \ldots, k$ the arc v_{i-1}, v_i exists in G^* and its cost is exactly $dist_G(v_{i-1}, v_i)$
  - Consider the graph G' just before v_i is contracted
  - Since $v_i < v_{i+1}$, both v_i and v_{i+1} are in that graph
  - By Lemma 1, we have $dist_{G'}(v_i, v_{i+1}) = dist_G(v_i, v_{i+1})$
  - The SP from v_i to v_{i+1} in G' can only use nodes $\geq v_i$
  - If that SP would have more than one arc, and the first arc would be v_i, w ... then w would have been our v_{i+1}
  - Hence the SP from v_i to v_{i+1} consist only of a single arc, and the cost of that arc is $dist_{G'}(v_i, v_{i+1}) = dist_G(v_i, v_{i+1})$

CH — Correctness Proof 6/6

- We are almost done
  - We have now proven that $\text{dist}_{G^{*}\uparrow}(s, v) = \text{dist}_G(s, v)$

    where v was the largest node on the SP from s to t
  - We can prove analogously that $\text{dist}_{G^{*}\downarrow}(v, t) = \text{dist}_G(v, t)$
  - Hence the SP cost will be amongst $\{\text{dist}_s[v] + \text{dist}_t[v] : v \in I\}$
  - By Lemma 1, $\text{dist}_{G^{*}}(s, t) = \text{dist}_G(s, t)$, that is, the cost of no shortest path decreases by adding shortcuts
  - Hence the query algorithm will compute exactly $\text{dist}_G(s, t)$

Node ordering 1/3



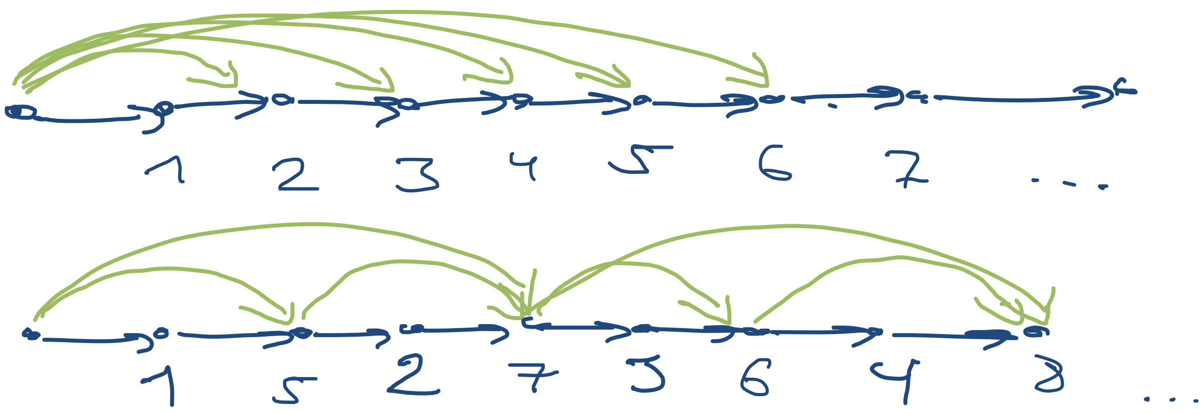
- General approach
  - Maintain the nodes in a priority queue, in the order of how **attractive** it is to contract the respective node next
  - Intuitively: the less shortcuts we have to add, the better
  - For each node, maintain the **edge difference** (ED):
    - S = the number of shortcuts that would have to be added if that node were contracted
    - E = the number of arcs incident to that node
    - Then the edge difference is simply ED = S – E
  - Note: when we contract a node, the edge difference of **any** node (not only the neighbours) may get affected

Node ordering 2/3

- How to maintain the ED for each node?
  - Initially compute the ED for each node (linear time)
  - Straightforward approach: recompute for **all** nodes after **each** single contraction → quadratic running time ... no good
  - Lazy update heuristic: update EDs "on demand" as follows:
    - Before contracting node with currently smallest ED, recompute its ED and see if it is still the smallest
    - If not pick next smallest one, recompute its ED and see if that is the smallest now; if not, continue in same way ...
  - Neighbours only heuristic: after each contraction, recompute EDs, but only for the neighbours of the contracted node
  - Periodic update heuristic: Full recomputation every x rounds

Node ordering 3/3

- Other criteria
  - Spatial diversity is also important, here is an example:



  - Spatial diversity heuristic: for each node maintain a count of the number of neighbours that have already been contracted, and **add** this to the ED
  - Note: the more neighbours have already been contracted, the later this node will be contracted

Implementation Advice 1/2

- Precomputation
  - Add arcs to the original graph, do **not** make a copy
  - Ignore arcs of already contracted nodes using arc flags
  - To compute the edge difference of a node, extend your contractNode method as follows:
    - add an argument bool computeEdgeDifferenceOnly
    - default is false; if true do the Dijkstras as usual, but in the end don't change anything in the graph, but just return the edge difference
  - To know which node to pick next, maintain all nodes in a priority queue, with key = edge difference

Implementation Advice 2/2

- Query algorithm
  - After the precomputation, set arc flags of all arcs u, v with u > v to true and all others to false
  - For the query algorithm, simply use Dijkstra with the considerArcFlags option (wrt the arc flags above)
    - one such Dijktra from the source, one from the target
    - compute $dist(s, t) = min\{dist_s[u] + dist_t[u]\}$ by a simple scan over the dist arrays from these two Dijkstras
    - as in the precomputation, avoid an $\Theta(\#nodes)$ reset of the dist arrays, but use the visitedNodes array instead
    - Note: no need to change any arc flags at query time!

Computing the actual path (not needed for Ex. Sheet)

- In the precomputation
  - When we contract a node v and add a shortcut u, w
    - then at that time dist(u, w) > cost(u, v) + cost(v, w)
    - Along with this shortcut, store the node v
    - Note: this is exactly one node per shortcut
  - In the query algorithm
    - first compute the SP in the upwards graph by backtracing parent pointers as usual (in each Dijkstra, both from the node on the SP with highest order)
    - then, while the paths contains a shortcut u, w replace it by u, v, w using the v stored above

References

- The CH paper again (for your convenience)

  Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks

  Geisberger, Sanders, Schultes, Delling, WEA 2008

  http://algo2.iti.uka.de/schultes/hwy/contract.pdf