Efficient Route Planning
SS 2012

Lecture 3, Wednesday May 9th, 2012
(A*, Landmarks, Set Dijkstra)

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 2
  - Public SVN snapshot of code from past exercises
- A* algorithm
  - A* = Dijkstra with a heuristic for goal direction
  - Heuristic 1: straight line distance
  - Heuristic 2: landmarks
- Exercise Sheet 3
  - Implement A* + run some queries for both heuristics
  - Some **optional** theoretical tasks (useful for exam preparation)

Your Feedback on Exercise Sheet 2

- Summary / excerpts last checked May 9, 15:57
  - Less work than last sheet, but still more than expected
  - 6 – 8 hours for most, but a few needed **much** longer
  - How to debug ... watch corresponding C++ lecture
  - Some spent quite some time on refactoring code from ES 1
  - Reduction to LCC was trickier than expected (node remapping)
  - "Not the first time I heard about Dijkstra"
  - Implementation advice from the lecture was useful again
  - Thanks a lot for the detailed advice from the tutor!
  - Result table is interesting + good correctness check
  - Problem with ant on Jenkins ... but fixed now
  - Machine to test code under equal conditions would be nice

Experimental results from Ex. Sheet 2

- See the table on the Wiki
  - #arcs in LCC is ≈ 99% of #arcs in original graph
  - #nodes is only ≈ 20% ... reason: **non**-highway nodes
  - Average query time is ≈ 1s for 1 million settled nodes
    - at least in C++, much slower in some Java implement.
  - On the avrg, ≈ 50% of all nodes settled per query ... a lot!
  - Average SP cost ≈ 1.5 hours on BaWü ... makes sense!
    - note that a random node does not give a random point on the map (many more nodes in city areas)

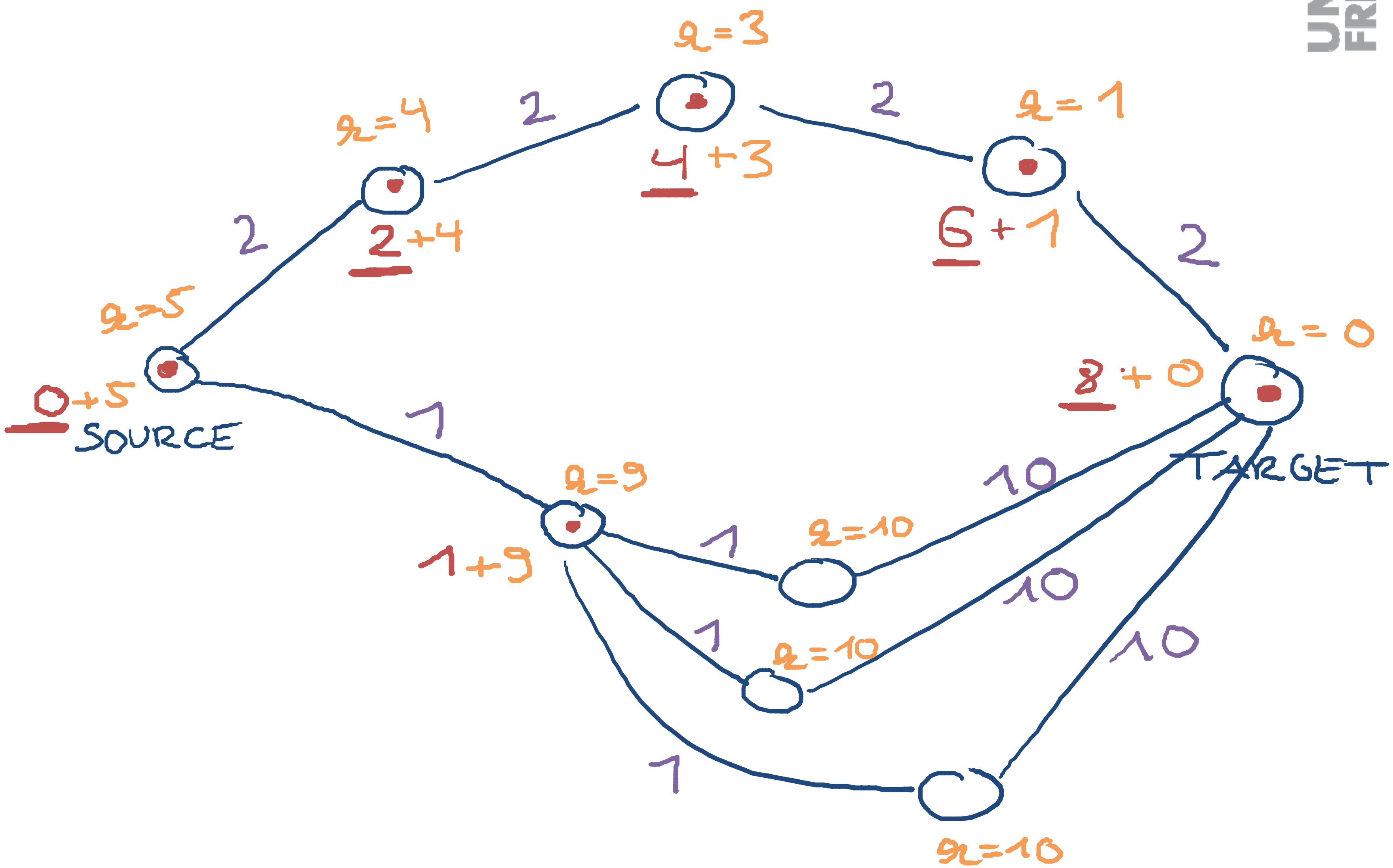
SVN snapshot

- SVN snapshot of code from past Exercise Sheets
  - After the deadline of an Exercise Sheet is over:
    - we will copy all **code** files from all user to a public subfolder snaphost of the course SVN ... see Wiki
  - That way you can
    - ... get inspiration from the code of others
    - ... continue even if you missed a previous Ex. Sheet
  - Feel free to ask if something is unclear in the code
    - For that purpose, please make sure that your code contains a copyright notice + email address

A* algorithm

- A* is a simple variant of Dijkstra's algorithm
  - Additionally: for each node u, a value h[u] that estimates dist(u, t), where t is the target
    - h is often called the heuristic function of A*
  - Difference to Dijkstra: value of a node u in the priority queue is not dist[u] but **dist[u] + h[u]**
    - therefore, if h[u] = 0 for all u, then A* = Dijkstra
  - Works if h is admissable and monotone ... later slide
  - Best results when h[u] = dist(u, t) for all u
    - then A* settles only the nodes on a shortest path

A* algorithm — Example

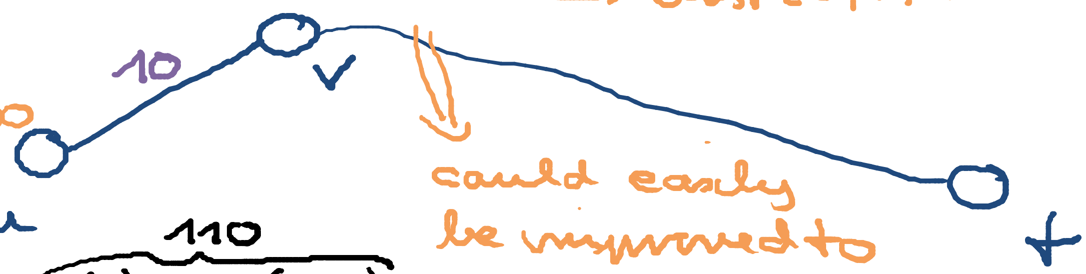


A* algorithm — Conditions on h

- The heuristic h must be **admissable**
  - For each node u it must hold: h(u) ≤ dist(u, t)
  - Informally: the heuristic must never overestimate
- The heuristic h must be **monotone**
  - For each arc (u,v) it must hold: h(u) ≤ cost(u,v) + h(v)
  - Informally: heuristic must obey the triangle inequality
  - Counterexample that shows why this is meaningful:

$h = 100 \Rightarrow dist(v,t) \geq 100$

$dist(u,t) \geq 200 \Leftarrow h = 200$



could easily be improved to 190

$\overbrace{h(u)}^{200} > \overbrace{h(v) + c(u,v)}^{110}$

NOT MONOTONE

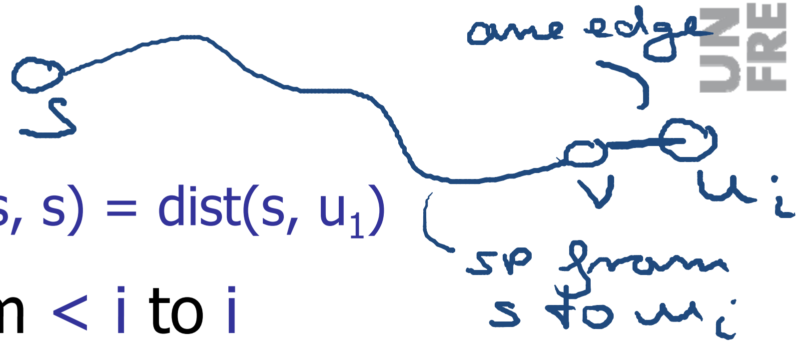
A* algorithm — Correctness proof 1/2

- Variant of the correctness proof for Dijkstra
  - Similarly as for Dijkstra, we make the simplifying assumption that the $dist(s, u) + h(u)$ are **all different**
  - Then we can order the nodes $u_1, u_2, u_3, \ldots$ such that

    $dist(s, u_1) + h(u_1) < dist(s, u_2) + h(u_2) < \ldots.$
  - Just as for Dijkstra we now prove by induction over i:

    In round i, node u_i is settled and $dist[u_i] = dist(s, u_i)$
  - Since we have already seen the proof for Dijkstra, we give this proof in condensed form ... verify for yourself!

A* algorithm — Correctness proof 2/2



- Case $i = 1$
  - In round 1, $dist[u_1] = 0 = dist(s, s) = dist(s, u_1)$
- Case $i > 1$: induction step from $< i$ to i
  - Let v be the direct predecessor of u_i on SP from s to u_i
    by MONOTONICITY
  - Then $dist(s,v) + h(v) \leq dist(s,v) + c(v,u_i) + h(u_i) =$
    $dist(s,u_i) + h(u_i)$ ... hence $v = u_j$ for some $j < i$
  - After v was settled and arc (v,u_i) relaxed in round $j < i$:
    RELAXATION INDUCTION HYPOTHESIS
    $dist[u_i] \leq dist[v] + c(v,u_i) = dist(s,v) + c(v,u_i) = dist(s,u_i)$
    BY NODE ORDERING
  - For $j > i$, $dist[u_j] + h(u_j) \geq dist(s,u_j) + h(u_j) > dist(s,u_i) + h(u_i)$
  - Hence u_i is settled in round i with $dist[u_i] = dist(s,u_i)$

A* algorithm — Two heuristics

- Straight-line distance (also: as-the-crow-flies distance)
  - Take $h(u) = eucl(u, t) / v_{max}$

    RECALL: $v = s/t$
    - where $eucl(u,t)$ is the **Euclidean** distance from u to t
    - and v_{max} is the maximum speed
  - Admissible and monotone because of triangle inequality
  - Optional theoretical exercise: verify this!
- Landmark heuristic
  - Informally: for every node u, precompute distances to a set of pre-selected nodes, called landmarks
  - How to obtain a heuristic function from that ... next slides

A* with landmarks 1/4

- Basic idea (first explained for **directed** graphs)
  - Consider an arbitrary node ℓ and call it a landmark
  - Our SP distance function dist satisfies the triangle inequality:

    $\text{dist}(u, v) \leq \text{dist}(u, w) + \text{dist}(w, v)$ for all nodes u, v, w
  - Then, in particular, for all landmarks ℓ and all nodes u, v

    $\text{dist}(u,\ell) \leq \text{dist}(u,v)+\text{dist}(v,\ell) \Rightarrow \text{dist}(u,\ell)-\text{dist}(v,\ell) \leq \text{dist}(u,v)$

    $\text{dist}(\ell,v) \leq \text{dist}(\ell,u)+\text{dist}(u,v) \Rightarrow \text{dist}(\ell,v)-\text{dist}(\ell,u) \leq \text{dist}(u,v)$
  - Hence, for a landmark ℓ, a target node t, and any node u

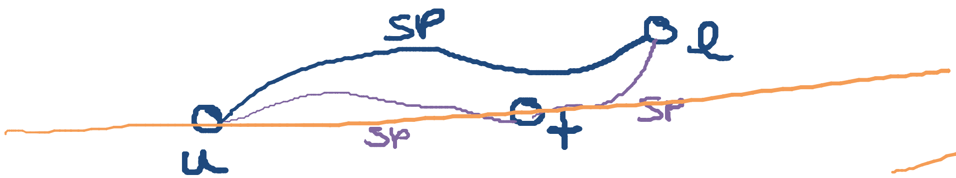
    $h(u) := \max(\text{dist}(u,\ell)-\text{dist}(t,\ell) , \text{dist}(\ell,t)-\text{dist}(\ell,u)) \leq \text{dist}(u,t)$
  - For undirected graphs, $\text{dist}(x,y) = \text{dist}(y,x)$ for all x,y and thus:

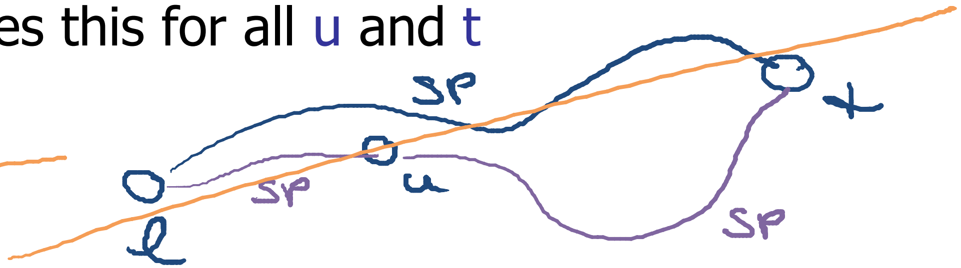
    $h(u) := |\text{dist}(\ell,u)-\text{dist}(\ell,t)| \leq \text{dist}(u,t)$

A* with landmarks 2/4

- When is this a good lower bound?
  - When one of these two inequalities is "close" to equality

    dist(u,ℓ) ≤ dist(u,t)+dist(t,ℓ) or dist(ℓ,t) ≤ dist(ℓ,u)+dist(u,t)
  - For the first inequality, this happens when t lies "close" to the shortest path from u to ℓ ... landmark "behind" target
  - For the second inequality, this happens when u lies "close" to the shortest path from ℓ to t ... landmark "before" u
  - Intuitively, landmark must be close to line through u and t
  - There is no ℓ which achieves this for all u and t





A* with landmarks 3/4

- Pick a **set** L of landmarks
  - For each $\ell \in L$ we have

    $\max(\ dist(u,\ell) - dist(t,\ell)\ ,\ dist(\ell,t) - dist(\ell,u)\ ) \leq dist(u,t)$
  - Hence also

    $\max_{\ell \in L}\{\max(dist(u,\ell) - dist(t,\ell)\ ,\ dist(\ell,t) - dist(\ell,u))\} \leq dist(u,t)$
  - When is the left hand side a good lower bound?
    - Obviously, the more landmarks the better
    - But for each landmark ℓ, we need to precompute and store $dist(u, \ell)$ and $dist(\ell, u)$ for all nodes u
    - Also, computing the lower bound at query time is $\sim |L|$
    - For a fixed number of landmarks, the more "distributed" over the graph they are the better

A* with landmarks 4/4

- Precomputation of landmark distances
  - We need $dist(u, \ell)$ and $dist(\ell, u)$ for all ℓ and u
  - Important: no need to do a Dijkstra for each u !
  - A **single** Dijkstra from ℓ gives us $dist(\ell, u)$ for **all** u
  - Similarly, a single Dijkstra on the graph with all arcs **reversed** gives us $dist(u, \ell)$ for all u
  - For our graphs, $dist(u, \ell) = dist(\ell, u)$ and the reversed graph is the same, and so a single Dijkstra per ℓ suffices
    - Heuristic is then $h(u) = \max_{\ell \in L} |dist(\ell, u) - dist(\ell, t)|$

Monotonicity of landmark heuristic

- Let (u,v) be an arbitrary arc with cost $c(u, v)$
  - We have to show that $h(u) \leq c(u,v) + h(v)$, where

    $h(u) = \max_{\ell \in L}\{\max(\text{dist}(u,\ell) - \text{dist}(t,\ell) , \text{dist}(\ell,t) - \text{dist}(\ell,u))\}$
  - For a fixed $\ell \in L$: $\text{dist}(u,\ell) \leq c(u,v) + \text{dist}(v,\ell)$ "triangle inequality"

    $\rightarrow$ $\text{dist}(u,\ell) - \text{dist}(t,\ell) \leq c(u,v) + \text{dist}(v,\ell) - \text{dist}(t,\ell)$ (1)
  - Similarly: $\text{dist}(\ell,v) \leq \text{dist}(\ell,u) + c(u,v)$

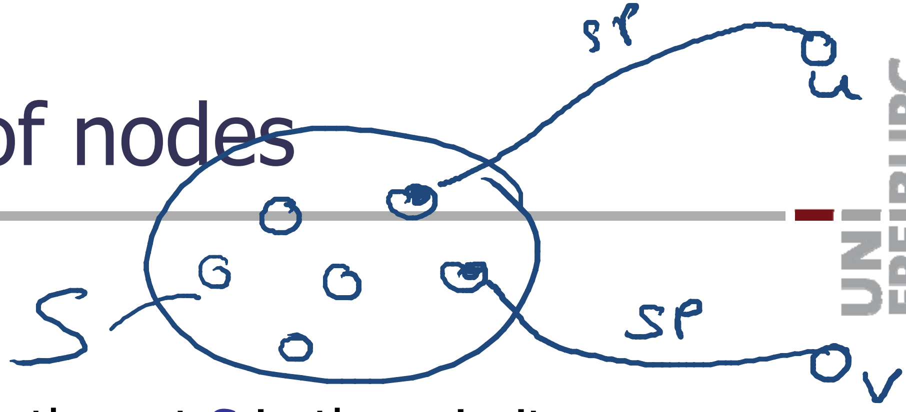
    $\rightarrow$ $\text{dist}(\ell,t) - \text{dist}(\ell,u) \leq c(u,v) + \text{dist}(\ell,t) - \text{dist}(\ell,v)$ (2)
  - Max of (1) and (2) gives us $h(u) \leq c(u,v) + h(v)$ for a single ℓ
  - If we then do $\max_{\ell \in L}$ on both sides, we are done
  - Lemma: if $x_i \leq y_i$ for all $i \in I$ ⇨ $\max_{i \in I} x_i \leq \max_{i \in I} y_i$

$$\max_{i \in I} x_i = \underset{\text{for some } j \in I}{x_j} \leq y_j \leq \max_{i \in I} y_i \quad \blacksquare$$

Landmark selection

- We look at two heuristics
  - Random selection
    - Not bad, but suboptimal distribution
  - Greedy farthest node selection
    - Start with a random node, then iteratively add more
    - In each iteration, pick the node u which **maximizes** **$\min_{\ell \in L'}$** $\mathrm{dist}(\ell, u)$, where L' = nodes already selected
      - intuitively: u is "farthest" from all nodes in L'
    - How to compute u with $\min_{\ell \in L'} \mathrm{dist}(\ell, u)$ for given L' ?

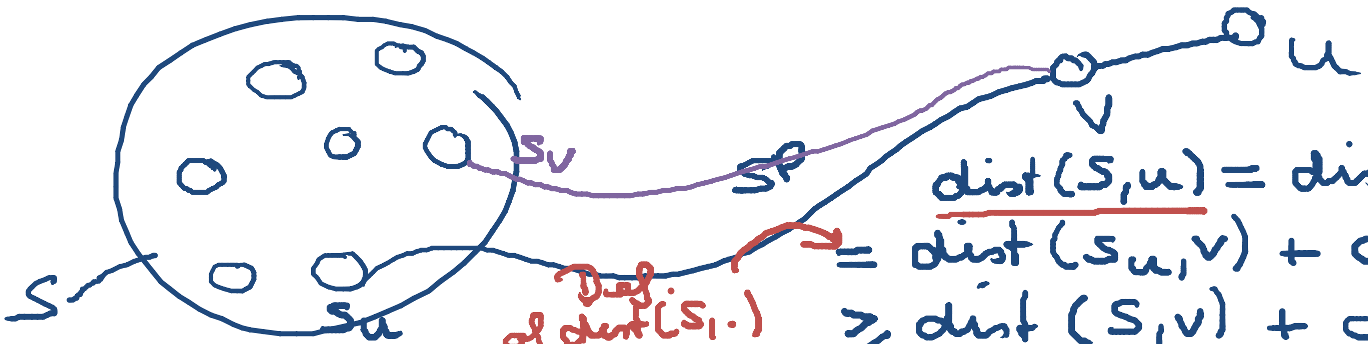
Dijkstra from a **set** of nodes



- Implementation
  - Initially put all nodes from the set S in the priority queue, with distance 0, then run ordinary Dijkstra
  - Then the distance computed for each node u will be

    $\min_{s \in S} \mathrm{dist}(s, u)$ ... which we write as $\mathrm{dist}(S, u)$
  - It's not obvious that this is true, so we should prove it
    - This is one of the **optional** exercises on Ex. Sheet 3.
    - Good thing to do when you prepare for the exam

$\mathrm{dist}(S,v) \le \mathrm{dist}(S,u)$ for $c > 0$



Def of $\mathrm{dist}(S,\cdot)$

$$\mathrm{dist}(S,u) = \mathrm{dist}(s_u, u)$$
$$= \mathrm{dist}(s_u, v) + c(v,u)$$
$$\ge \mathrm{dist}(S, v) + c(v,u)$$

A* — Implementation advice

- No need to implement a new class
  - You can easily extend your class DijsktrasAlgorithm
  - Just add a member variable Array<int> heuristic ... see Wiki
- Landmark precomputation
  - Important: **don't** execute one Dijsktra for each node
  - For undirected graphs, one Dijkstra per landmark suffices
    - for each ℓ, this gives you $\text{dist}(\ell, u)$ for all u
    - heuristic is $h(u) = \max_{\ell \in L}\{|\text{dist}(\ell, u) - \text{dist}(\ell, t)|\}$
  - Note that the heuristic h must be computed **per query**
    - for simplicity, for a given query, first compute $h(u)$ for **all** nodes u ... see design suggestion linked from Wiki

References

- The original "A* with landmarks" paper

  Computing the shortest path: A* search meets graph theory

  A. Goldberg and C. Harrelson, SODA 2005

  http://portal.acm.org/citation.cfm?doid=1070432.1070455

  http://www.avglab.com/andrew/pub/soda05.pdf