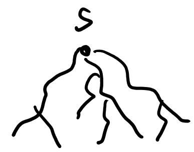
08 A*

Tuesday, February 4, 2025 13:52

Dijkstra:
- single source shortest paths
- shortest path tree
- $O(|E| \log |E|)$



shortest paths from s to every other vertex

What if we only want one particular target $t \in V$? Can we get a faster algorithm?
Given $s, t \in V$, $G = (V, E)$ compute $\text{dist}(s,t)$ (or the shortest path from s to t).

```
Dijkstra(s)
  queue.add(0, s, t)
  while queue not empty
    d,v = queue.pop()
    if v==t then return d
    if visited[v]
        continue
    visited[v] = true
    distance[v] = d   // dist(s,v)
    for (v,u) ∈ E
        queue.add(d + c_vu, u)
```

All $v \in V$ with $\text{dist}(s,v) < \text{dist}(s,t)$ will be visited before t.

$\Rightarrow$ many visited vertices
$\Rightarrow$ algorithm takes a lot of time.

Idea: Can we shift the priorities?
- Prioritize vertices that are (likely) on the right direction
- Deprioritize vertices that are (likely) in the wrong direction

---

```
A-star(s,t)  // A* with visited check
  queue.add(0, 0, s)
  while queue not empty
    p, d,v = queue.pop()
    if v==t then return d
    if visited[v]
        continue
    visited[v] = true
    distance[v] = d   // dist(s,v)
    for (v,u) ∈ E
```

Idea: $h(u)$ is large then u is deprioritized
$h(u)$ is small then u is prioritized

h is called heuristic

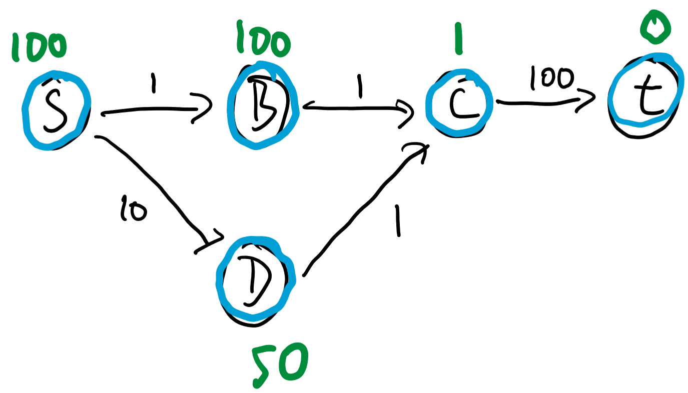
$h(u) \approx \text{dist}(u,t)$ ← some estimate for the distance

distance[v] = d // dist(s,v)
for (v,u) ∈ E
queue.add($d + c_{vu} + h(u)$, $d + c_{vu}$, u)
Priority distance

$h(u) \approx dist(u,t)$ ... for the distance

Example: $h(u) = |u.x - t.x| + |u.y - t.y|$

Issue: Correctness proof of Dijkstra required priority = distance.
Not satisfied!
Does Algo still work?



Queue	prio	dist
~~S~~	~~0~~	~~0~~
~~B~~	~~101~~	~~1~~
~~D~~	~~60~~ (10+50)	~~10~~
~~C~~	~~12~~	~~11~~
t	111	111
~~C~~	~~3~~	~~2~~ ← 2 < 11
~~t~~	~~102~~	~~102~~

A-star(s,t) // A* without visited check
queue.add(0, 0, s)
while queue not empty
p, d, v = queue.pop()
if v == t then return d
if ~~visited[v]~~ distance[v] < d
continue
~~visited[v] = true~~
distance[v] = d // dist(s,v)
for (v,u) ∈ E
queue.add($d + c_{vu} + h(u)$, $d + c_{vu}$, u)
Priority distance

Def: h is called admissible if

Def: h is called admissible if

$$h(u) \leq dist(u,t) \quad \forall u \in V$$

Thm: If h is admissible then A^* without visited check will find the shortest path.

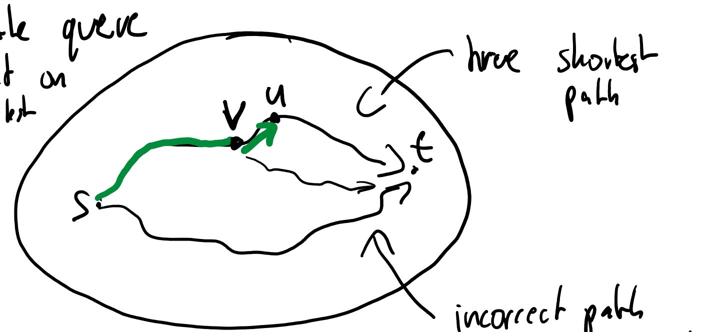
Proof: Assume distance[t] is wrong even though h is admissible

Let v be the last vertex on the correct shortest path that we visited before t

v was visited $\Rightarrow$ all neighbors of v are in the queue

Let u be the neighbor of v that is next on the shortest path

1) u is still in queue

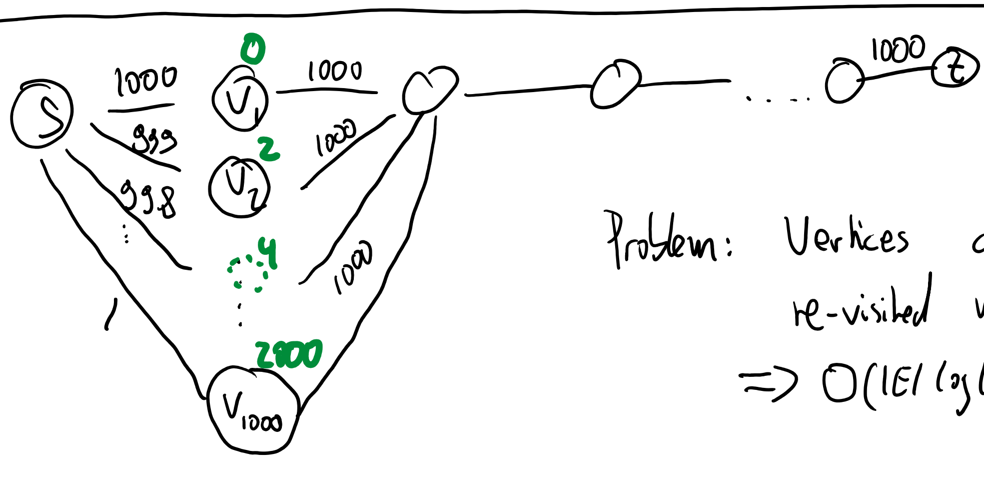


$$\begin{aligned} \underset{\overset{\vee}{dist(s,t)}}{prio(t)} < prio(u) &= distance[v] + c_{vu} + h(u) \\ &= dist(s,u) + h(u) \\ &\leq dist(s,u) + dist(u,t) \\ &= dist(s,t) \quad ↯ \end{aligned}$$

2) u is not in queue anymore

$\Rightarrow$ we visited u

$\Rightarrow$ v is not last visited vertex ↯

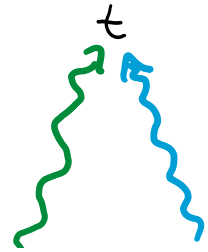


Problem: Vertices could be re-visited multiple times

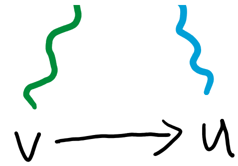
$\Rightarrow O(|E| \log |E|)$ does not hold anymore

Def: h is consistent if

$$h(v) \leq h(u) + c_{vu} \quad \forall (v,u) \in E$$



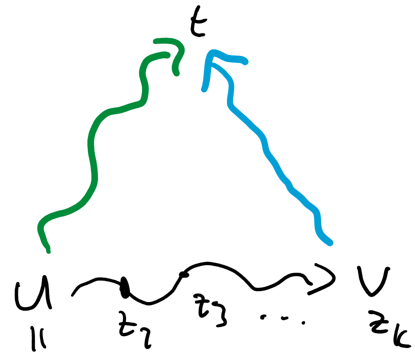
$$h(v) \le h(u) + c_{vu} \qquad \forall (v,u) \in E$$



Thm: If h is consistent then A^* <u>with</u> visited check finds the shortest path.
$(\Rightarrow O(|E| \log |E|)$ worst-case holds again$)$

Lemma: If h is consistent
Let $P \subseteq E$ be a path from u to v

$$h(u) \le \text{length}(P) + h(v)$$



$$h(z_1) \le h(z_2) + c_{z_1 z_2} \le \left(h(z_3) + c_{z_2 z_3}\right) + c_{z_1 z_2}$$
$$\vdots$$
$$\le h(z_k) + \sum_{i=1}^{k-1} c_{z_i z_{i+1}}$$

		v. check	v. no check
h	admissible	✗	✓
	consistent	✓	✓