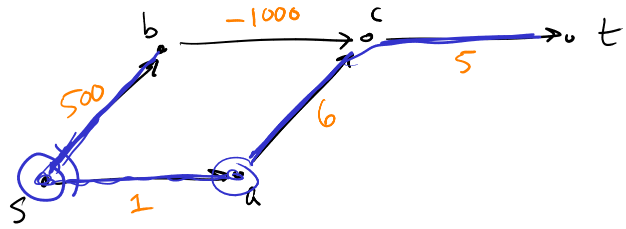
Greedy algorithms:

- At each step, make a locally best choice
- Leads to a globally best solution




Dijkstra's Algorithm

Edsger Dijkstra — 1956

AMRAC ?

**Def'n** A weighted, (directed) graph is a graph where each edge has an associated weight. We denote the weight of an edge $(u, v)$ by $w_{uv}$ or $w(u, v)$. The weight of a path is the sum of the weights along the path.

For now, we will consider weights in $\mathbb{R}_{\geq 0}$ (nonnegative real #'s).

Later we will consider $\mathbb{R}$.

What problems does Dijkstra solve?

1. Find the shortest path $s \to t$ in a weighted graph.
2. Find the shortest paths from $s$ to every other vertex in a weighted graph. (Single-Source Shortest Path, SSSP)

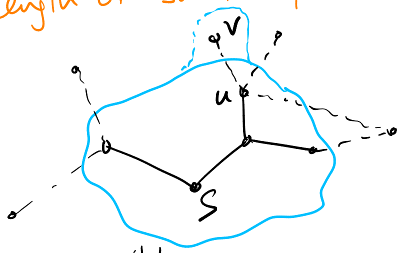
---

| BFS | | Dijkstra |
|---|---|---|
| layer dict | —— | distance dict (i.e. weight of shortest path) |
| parent dict | —— | same. |
| visited set | —— | same. |
| queue of vertices | | Friday. |

BASICDIJKSTRA(G, s):

Mark s VISITED, all others UNVISITED

d ← empty dict — d[v] = length of shortest path s → v.




d[s] ← 0

while not all vertices are VISITED:

Pick VISITED u, UNVISITED v such that d[u] + w(u,v) is as small as possible. — i.e. v is vertex water will reach next.

Mark v VISITED.

d[v] ← d[u] + w(u,v)

parent[v] ← u.

return d, parent.

---

Theorem. Dijkstra correctly solves SSSP problem, i.e. d[v] will be the length of the shortest path s → v for all v.

Proof. As our loop invariant, we will choose the proposition "d[v] is the length of shortest s → v path for all VISITED v".

- The loop invariant holds before the loop executes because the only visited vertex is s, and d[s] = 0, which is indeed the shortest distance from s to itself (since all weights are ⩾ 0).
- Now suppose the invariant holds, we must show it still holds after executing the loop one more time.