Dijkstra's Algorithm

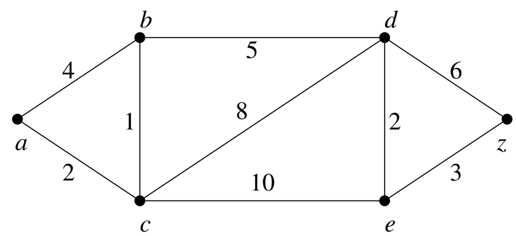
Problem. Given a connected network $G = (V, E)$; $w\colon E \to \mathbb{R}_{\geq 0}$ and two vertices $a, z \in V$, find the w-shortest path from a to z.

Dijkstra's Algorithm. Throughout, we'll have a decomposition $V = C \sqcup U$ of V into two disjoint sets, C for Confirmed / Confident and U for Unknown, a function $m\colon V \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ for "min so far", and a partially defined function $b\colon V \to V$ for "backtracking". As we run, the set C will grow and the set U will shrink. We stop when $z \in C$, and then $m(z)$ will be "the time to z" and $(z, b(z), b(b(z)), \ldots)$ will be the path from a to z, going backwards.

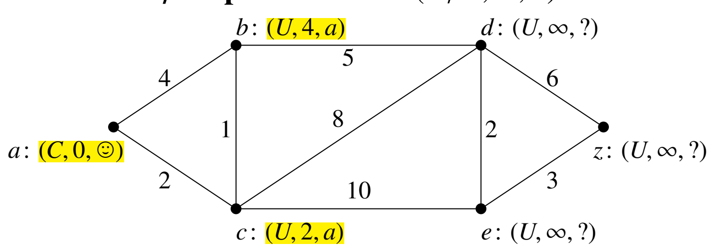
Initialization. Set $C := \{a\}$, $U := V \setminus \{a\}$, $m(a) := 0$, and $b(a) := ☺$. Also, if x is adjacent to a set $m(x) := w(ax)$ and $b(x) := a$, and for all other x set $m(x) := \infty$ and $b(x) = \text{undef}$.

Iteration. Let x_0 be where m attains its minimum on U (breaking ties arbitrarily), move x_0 from U to C, and for each neighbor $y \in U$ of x_0, if $m(x_0) + w(x_0 y) \geq m(y)$, do nothing. Else set $m(y) := m(x_0) + w(x_0 y)$ and $b(y) := x_0$.

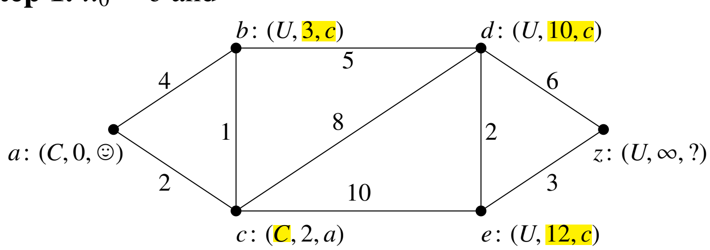
Example. Solve the shortest path problem for the network:



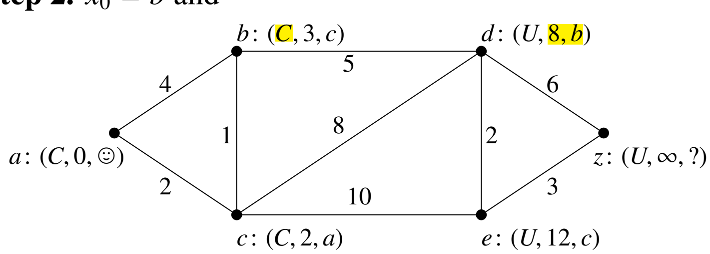
Initialization / Step 0. Notation: $(C/U, m, b)$



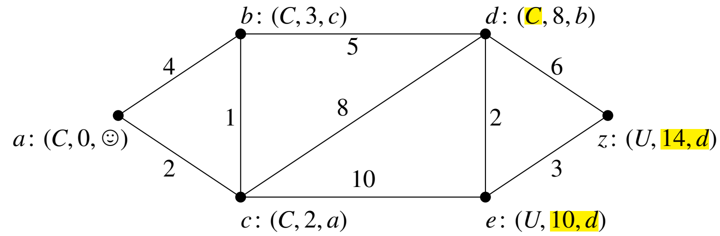
Step 1. $x_0 = c$ and



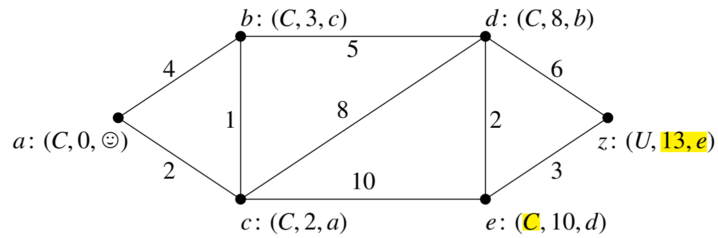
Step 2. $x_0 = b$ and



Step 3. $x_0 = d$ and



Step 4. $x_0 = e$ and



Step 5, end. $x_0 = z$ so we are done; the minimal path length is $m(z) = 13$, and the path, going backwards, is $z \xrightarrow{b} e \xrightarrow{b} d \xrightarrow{b} b \xrightarrow{b} c \xrightarrow{b} a$.

Proof that the algorithm works. The inductive assertion is "after each step, for each $x \in C$ the minimal path length to x is $m(x)$ and the stop before x is $b(x)$; for each $x \in U$ for which $m(x) < \infty$, the minimal path length where all stops but the last are in C is $m(x)$ and the stop before x (in such a path) is $b(x)$".

Efficiency estimate. For concreteness, take $|V| = 1{,}000{,}000$ and assume that the maximal degree of a vertex is 7.

Very naively, the search for x_0 is fast and we need about $7{,}000{,}000$ operations in total.

Less naively, the search for x_0 takes about $500{,}000$ operations, so our total is $1{,}000{,}000 \times (7 + 500{,}000)$.

Cleverly, instead of a search, we maintain an ordered table of the values of m. Updating the table takes about $\log_2(500{,}000) \sim 20$ operations, so the total number of operations required is about $1{,}000{,}000 \times (7 + 7 \times 20)$, a feasible number.

Finding a minimal spanning tree.

Kruskal's Algorithm. Start with $T = \emptyset$; repeatedly add to T the cheapest edge that does not form a circuit with edges already in T.

Prim's Algorithm. Start with $T = \emptyset$; repeatedly add to T the cheapest edge that connects T and the complement T^c of T.

Read Along. Section 4.1 and 4.2, and all of section 5.
HW5 will be on the web by midnight tonight, October 29.
No Dror office hours on Tuesday November 3, sorry.