Searching Graphs

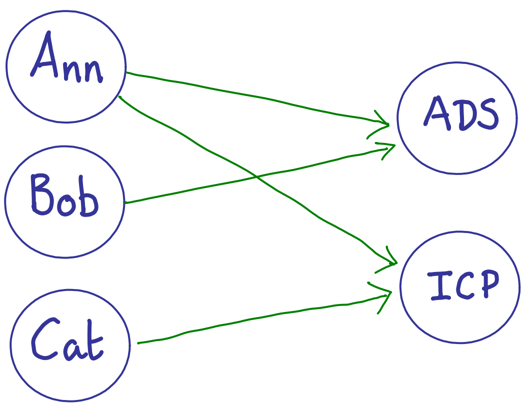
Example : London underground

Graph	Relation
undirected	symmetric
colour	union
path	reflexive, transitive closure

S, T — sets (usually finite in examples)

Definition A *relation of type* $S \sim T$ is a subset of $S \times T$ (the set of pairs (s,t) where $s \in S \wedge t \in T$).

Example takes of type Student $\sim$ Module .

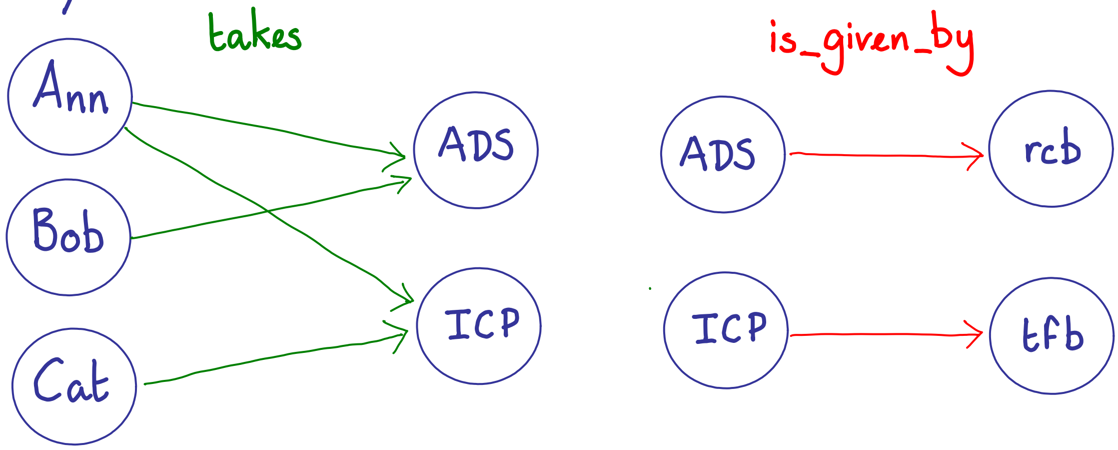


"bipartite graph"

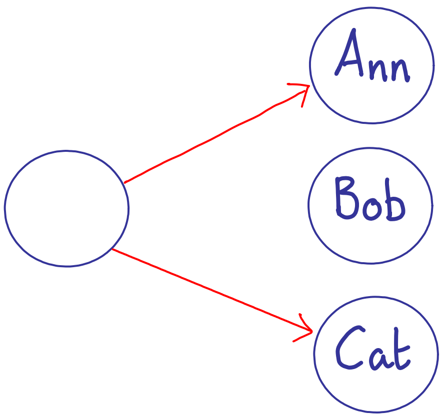
Definition If X is a relation of type $S \sim T$ and Y is a relation of type $T \sim U$, the *composition* $X \bullet Y$ is the relation of type $S \sim U$ defined by

$$X \bullet Y = \{ s,t,u : (s,t) \in X \wedge (t,u) \in Y : (s,u) \} .$$

Example



A subset S of a universe $\mathbb{U}$ can be viewed as a relation of type $\mathbb{1} \sim \mathbb{U}$ where $\mathbb{1}$ is a one-element set.

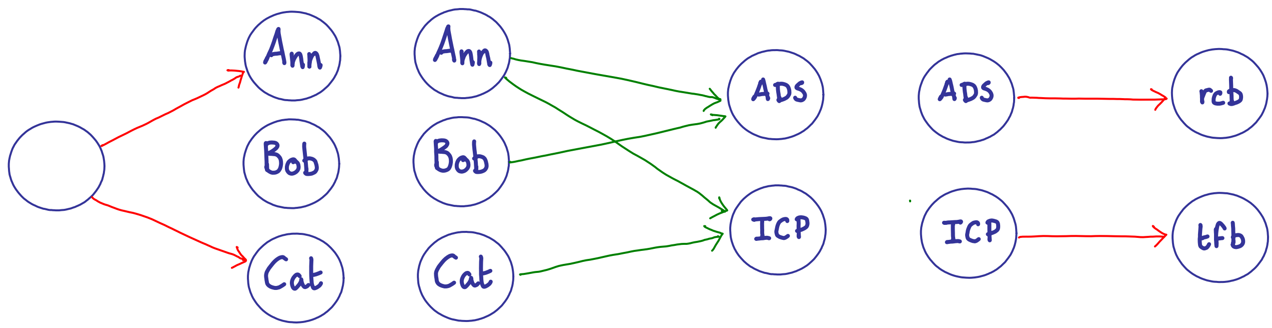


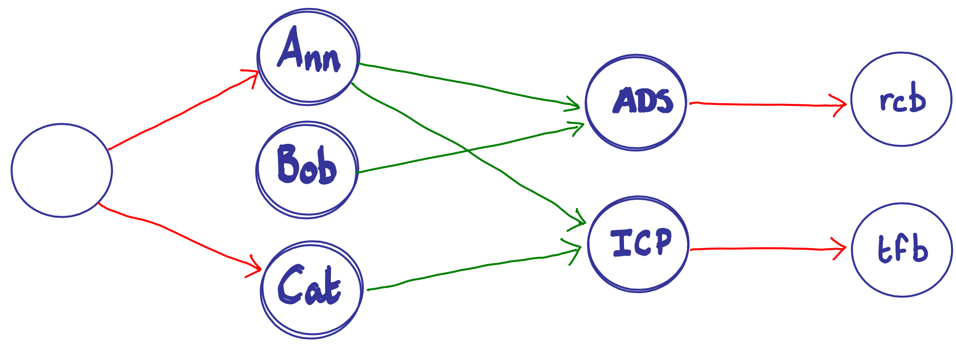
{Ann,Cat}

Write: $\langle S |$

Composition of relations is associative

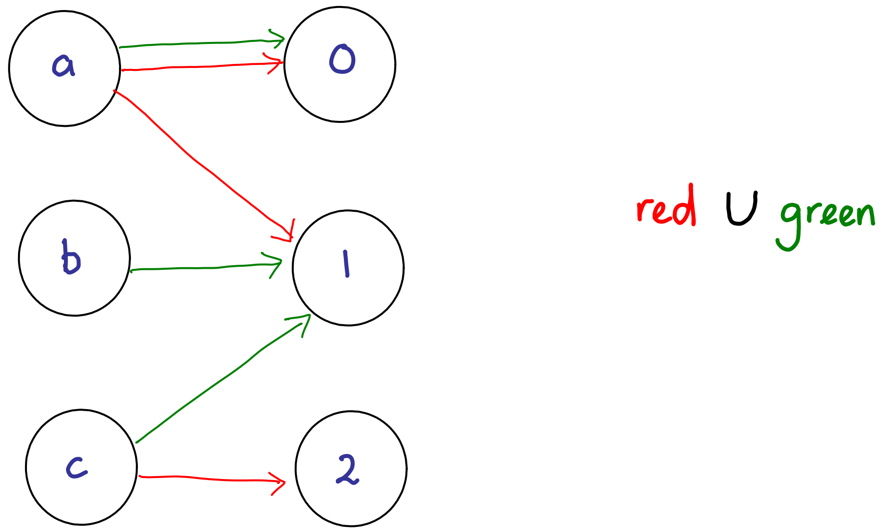
$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$





Set of lecturers taken by Ann and Cat.

Relations are sets. The union of two relations is meaningful if they have the same type.



Composition distributes through union.

$$X \cdot (Y \cup Z) \;=\; X \cdot Y \cup X \cdot Z$$

$$(X \cup Y) \cdot Z \;=\; X \cdot Z \cup Y \cdot Z \quad .$$

The empty relation is the zero of composition.

$$\emptyset \cdot X \;=\; \emptyset \;=\; X \cdot \emptyset$$

The identity relation (of appropriate type) is its unit

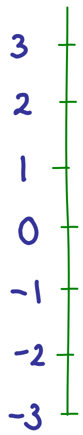
$$Id_S \cdot X \;=\; X \;=\; X \cdot Id_T \qquad (X : S \sim T)$$

$$Id_S \;=\; \{x : x \in S : (x,x)\} \quad .$$

Binary Relations

A *binary* relation is a relation of type $S \sim S$ for some S.

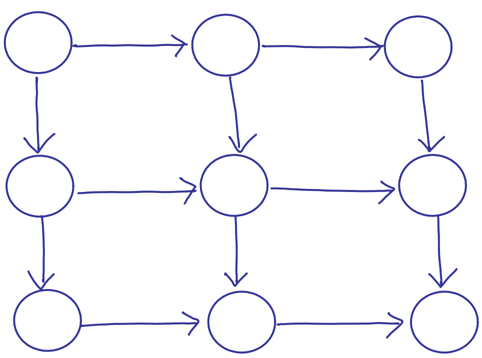
Example Predecessor relation



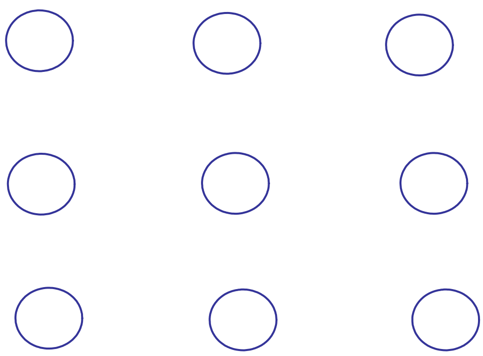
$$m \text{ predecessor } n \;\equiv\; m = n-1$$

(note convention when displaying graph).

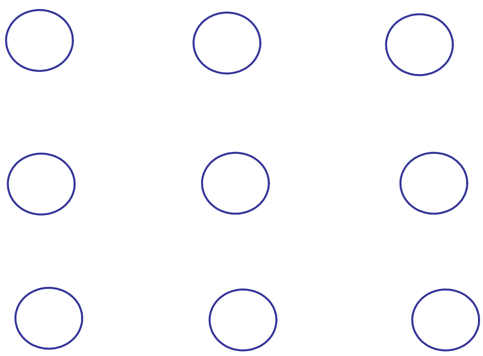
X



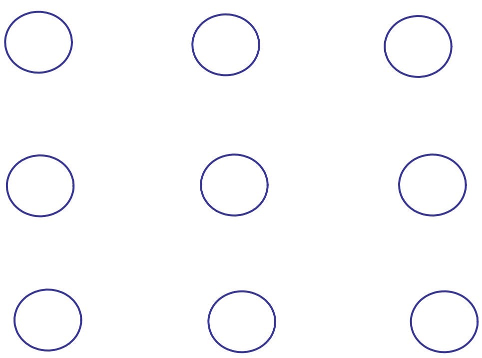
X^2



X^3



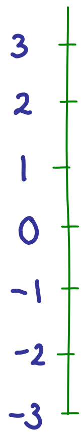
X^0



$$X^* = \langle \cup k : 0 \le k : X^k \rangle$$

X^* is the *reflexive, transitive closure* of X.

Example



$$\text{at\_most} = \text{predecessor}^*$$

Id identity relation of same type as X — $\{x :: (x,x)\}$

$$X^* = \text{Id} \cup X \cdot X^* = \text{Id} \cup X^* \cdot X$$

$\text{Id} \subseteq X^*$ (reflexive)

$X^* \cdot X^* \subseteq X^*$ (transitive)

$\langle \forall Y, Z : \quad Y \cup Z \cdot X \subseteq Z \Rightarrow Y \cdot X^* \subseteq Z \rangle$

(closure)

Graph Searching: given a set S and a relation R, determine $\langle S| \cdot R^*$

(nodes reachable from nodes in S by paths such that edges satisfy R).

V finite set of *nodes*
G directed graph/binary relation on V.
S subset of V (*start* nodes)

Basic algorithm:

$[\![\quad n : \mathbb{1} \sim V$

$\triangleright \quad n := \langle S| \;;$

$\{$ Invariant : $\langle S| \cdot G^* = n \cdot G^*$

Bound fn. : $|V| - |n|$ $\}$

do $n \neq n \cup n \cdot G \rightarrow n := n \cup n \cdot G$
od

$\{ \quad n = \langle S| \cdot G^* \quad \}$

$]\!]$

Basic algorithm:

$[\![\quad n : \mathbb{1} \sim V \;; \quad k : \mathbb{N} \;;$ /* auxilliary variable */

$\triangleright \quad n := \langle S| \;; \quad k := 0 \;;$

$\{$ Invariant : $n = \langle S| \cdot G^{\leq k}$

Bound fn. : $|V| - |n|$ $\}$

do $n \neq n \cup n \cdot G \rightarrow k, n := k+1, n \cup n \cdot G$
od

$\{ \quad n = \langle S| \cdot G^* \quad \}$

$]\!]$

$G^{\leq k} = \langle \cup j : 0 \leq j \leq k : G^j \rangle$.

Uses $G^* = G^{\leq |V|}$ for finite V and $G : V \sim V$.

b "black" nodes
g "grey" nodes

$$[\![\quad b, g : 1\!\!1 \sim V$$

$$\triangleright \quad g := \langle S| \;;\quad b := \emptyset \;;$$

$$\{ \text{Invariant} : \quad \langle S| \cdot G^* = b \cup g \cdot G^*$$

$$\text{Bound fn.} : \quad |V| - |b| \qquad \}$$

$$\mathbf{do} \quad g \neq \emptyset \rightarrow b, g := b \cup g,\; g \cdot G \cap \neg(b \cup g)$$
$$\mathbf{od}$$

$$\{ \; b = \langle S| \cdot G^* \; \}$$

$$]\!]$$

Avoiding recomputation

Elementwise implementation :

$$b, g := \emptyset, S \;;$$

$$\{ \text{Invariant} : \quad \langle S| \cdot G^* = \langle b| \cup \langle g| \cdot G^*$$

$$\text{Bound fn.} : \quad |V| - |b \cup g| \qquad \}$$

$$\mathbf{do} \; \langle \square v : \; v \in g$$
$$: \; b, g := b \cup \{v\},\; (g - \{v\}) \cup (\langle v| \cdot G \cap \neg(b \cup g))$$
$$\rangle$$
$$\mathbf{od}$$

$$\{ \; g = \emptyset \wedge \langle S| \cdot G^* = \langle b| \; \}$$

Breadth-first search : implement g as a queue

Depth-first search : implement g as a stack .

Topological Search

Assumes G is acyclic. Equivalently $G^{|V|} = \emptyset$.

Constructs a total ordering of the nodes of G.

Postcondition:

$$\langle \forall u, v :: \langle u | \cdot G^+ \cdot | v \rangle \Rightarrow f.u < f.v \rangle$$

(G^+ is the transitive closure of G.

$G^+ = \langle \cup k : 1 \leq k : G^k \rangle$.)

b "black" nodes
g "grey" nodes

$G\restriction b$ graph G restricted to nodes in b

$\langle u|\cdot G\restriction b\cdot|v\rangle \;\equiv\; u\in b \wedge v\in b \wedge \langle u|\cdot G\cdot|v\rangle$

$\{\; G^{|V|} = \emptyset \;\}$

for $v\in V$ **do** $in.v := |\{w \mid (w,v)\in G\}|$ **rof** ; /* "in degree" */

$b := \emptyset$; $g := \{v \mid in.v = 0\}$; $k := 0$;

$\{$ Invariant : $\langle \forall u,v : u\in b \wedge v\in b$
$\quad\rangle \quad : \quad \langle u|\cdot(G\restriction b)^+\cdot|v\rangle \Rightarrow f.u < f.v \leqslant k \;\}$

do
$\quad\langle \square v : \; v\in g \wedge in.v = 0$
$\qquad : \; b,g := b\cup\{v\},\, g-\{v\}$; $f.v, k := k, k+1$;
$\qquad\quad$ **for** $w\in\langle v|\cdot G$ **do** $g, in.w := g\cup\{w\}, in.w-1$
$\qquad\quad$ **rof**
$\quad\rangle$
od

$\{\; \langle \forall u,v :: \langle u|\cdot G^+\cdot|v\rangle \Rightarrow f.u < f.v\rangle \;\}$

Additional invariants:

$\langle \forall v : \neg(v\in b\cup g) \; : \; 0 < in.v\rangle$

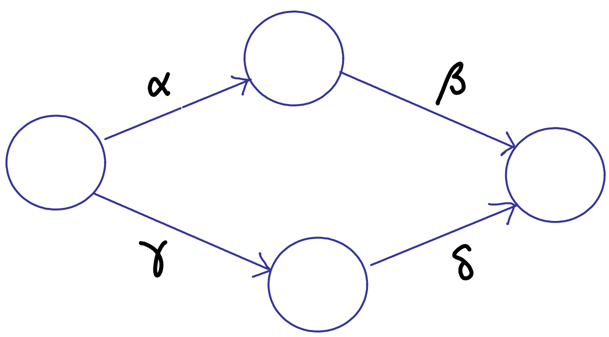
$g\neq\emptyset \;\Rightarrow\; \langle \exists v : v\in g : 0 = in.v\rangle$

(assumes acyclicity of G).

Shortest Paths

focus on

- understanding the function of each of the program variables by understanding the invariants
- the algebra of operations on graphs .



Edge labels are *symbols*.

d maps edge labels to distances.

Extend d to paths:

and to sets of paths:

D distance graph/matrix

$$\langle u| \cdot D \cdot |v\rangle \;=\; \begin{array}{ll} \textbf{if} & (u,v) \in G \rightarrow d.(u,v) \\ \Box & (u,v) \notin G \rightarrow \infty \\ \textbf{fi} & \end{array}$$

$\langle u|$ $1 \times |V|$ graph/matrix with one edge (to u) of distance 0

$|v\rangle$ $|V| \times 1$ graph/matrix.

Definitions

$D \downarrow E$ is given by

$$\langle u| \cdot D{\downarrow}E \cdot |v\rangle \;=\; \langle u| \cdot D \cdot |v\rangle \downarrow \langle u| \cdot E \cdot |v\rangle$$

$D \cdot E$ is given by

$$\langle u| \cdot D \cdot E \cdot |v\rangle \;=\; \langle \Downarrow w :: \langle u| \cdot D \cdot |w\rangle + \langle w| \cdot E \cdot |v\rangle \rangle$$

(where $d + \infty = \infty$)

$$D^* \;=\; \langle \Downarrow k : 0 \leq k : D^k \rangle \quad .$$

Shortest distance from u to v

$$\langle u| \cdot D^* \cdot |v\rangle \quad .$$

Observe: for labelled graphs G, H:

$$d.(G \cup H) = d.G \downarrow d.H$$

$$d.(G \bullet H) = d.G \cdot d.H$$

$$d.(G^*) = (d.G)^*$$

Also, for distance graphs D, E, F

$\downarrow$ is associative, symmetric and idempotent

$\cdot$ is associative

$$D \cdot (E \downarrow F) = (D \cdot E) \downarrow (D \cdot F)$$

(Dijkstra's) Wave Algorithm

b "black" nodes
g "grey" nodes

$b, g := \emptyset, \{S\}$; for $u \in V$ do $d.u := \infty$ rof ; $d.S := 0$;

{ Invariant: $\langle \forall u : u \in b : \langle S| \cdot D^* \cdot |u\rangle = d.u \rangle$

$\quad \wedge \langle \forall u : u \notin b : \langle S| \cdot D^* \cdot |u\rangle = \langle \Downarrow v : v \in g : d.v + \langle v| \cdot D^* \cdot |u\rangle \rangle\rangle$

}

do $\langle \Box v : v \in g \wedge d.v = \langle \Downarrow u : u \in g : d.u \rangle$

$\quad : b := b \cup \{v\}$; $g := g - \{v\}$;

$\quad\quad$ foreach edge (v,w) from v do

$\quad\quad\quad d.w := d.w \downarrow (d.v + \langle v| \cdot D \cdot |w\rangle)$;

$\quad\quad\quad g := g \cup (\{w\} - (b \cup g))$

$\quad\quad$ rof

$\rangle$

od

{ $g = \emptyset \wedge \langle S| \cdot D^* = \underset{\sim}{d}$ }

Implementation Problems

1. Initialisation:

   for $u \in V$ **do** $d.u := \infty$ **rof** ;

2. Choice of v :

   $v \in g \;\wedge\; d.v = \langle \Downarrow u : u \in g : d.u \rangle$

Standard Solution : use "priority queue" for g.

Observe :

each edge is "visited" exactly once

for each edge (v,w) from v **do**
$d.w := d.w \downarrow (d.v + \langle v| \cdot D \cdot |w \rangle)$;
$g := g \cup (\{w\} - (b \cup g))$
rof

Idea :

compute *edge* distances

choose closest grey *edge* .

```
|[ b : set of node ;   /* "black" nodes */
   g : set of edge ;   /* "grey" edges */
   d : node → distance ;   /* undefined for "white" nodes */
   ed : edge → distance ;   /* undefined for "white" edges */
 ▷ b := {S} ;  g := {e | e∈edge ∧ from.e = S }
   for e ∈ g do ed.e := ⟨from.e |·D·| to.e⟩ rof
   { Invariant :     ⟨∀u: u∈b: ⟨S|·D*·|u⟩ = d.u ⟩
         ∧ ⟨∀u: u∉b: ⟨S|·D*·|u⟩ = ⟨⇓e: e∈g: ed.e + ⟨to.e|·D*·|u⟩⟩
         ∧ ⟨∀e: e∈g: ed.e = ⟨S|·D*·|from.e⟩ + ⟨from.e |·D·| to.e⟩⟩
   }
   do F∉b ∧ g≠∅ → if( □e : e∈g ∧ ed.e = ⟨⇓e′: e′∈g: ed.e′⟩ :
                        v := to.e ;  g := g-{e} ;
                        if v∈b → skip
                        □ v∉b → b := b∪{v} ;  d.v := ed.e ;
                                 for each e′ s.t. from.e′ = v
                                    do ed.e′ := d.v + ⟨v|·D·|to.e′⟩
                                 rof
                        fi
                  ) fi
   od
{(F∈b ∧ d.F = ⟨S|·D*·| F⟩) ∨ (F∉b ∧ d.F = ∞) }
]|
```

```
b := {S} ;  g := {e | e∈edge ∧ from.e = S }
for e ∈ g do ed.e := ⟨from.e |·D·| to.e⟩ rof
```

Assumption: edge distances are natural numbers less than MAX

```
b := {S} ;   g := {e | e∈edge ∧ from.e = S }
for e ∈ g do ed.e := ⟨from.e |·D·| to.e⟩ rof
dist := 0;
{Invariant: ⟨∀u: u∈b: d.u ≤ dist⟩
            ∧ ⟨∀e: e∈g: dist ≤ ed.e < dist + MAX⟩ }
do F∉b ∧ g≠∅ →
        if ⟨□ e: e∈g ∧ dist = ed.e
                v := to.e ;  g := g-{e} ;
                if v∈b → skip
                □ v∉b → b := b∪{v} ;  d.v := dist
                        for each e' s.t. from.e' = v
                            do ed.e' := dist + ⟨v|·D·|to.e'⟩
                        rof
                fi
        ⟩
        □ otherwise → dist := dist + 1
        fi
od
```

Implementation of g

gs : array$[0..MAX)$ of Stack of edge ;

Invariant :

$$g = \langle \cup i : 0 \le i < MAX : Setify.(gs[i]) \rangle$$

$$\wedge \ \langle \forall i,e : e \in gs[i] : ed.e = dist + (i - dist) \bmod MAX \rangle$$

$e \in g \wedge dist = ed.e$: $\quad e := pop.(gs[dist \bmod MAX])$

$ed.e' := dist + \langle from.e' |\cdot D \cdot| to.e' \rangle$:

$$push.(gs[(dist + \langle from.e' |\cdot D \cdot| to.e' \rangle) \bmod MAX], e')$$

A^* Algorithm

- Given start, S, and finish, F, find

  $\langle S| \cdot D^* \cdot |F\rangle$

- Assumes an "admissible heuristic" $\underset{\sim}{H}$:

  $\underset{\sim}{H} \leqslant D \cdot \underset{\sim}{H}$

  (i.e. $\langle \forall u,v :: H.u \leqslant \langle u| \cdot D \cdot |v\rangle + H.v \rangle$)

  For example, crow-fly distance to finish.

- It's usual to require that $H.F = 0$ but, for correctness, this is unnecessary.

---

b "black" nodes
g "grey" nodes

$b,g := \emptyset, \{S\}$; **for** $u \in V$ **do** $d.u := \infty$ **rof** ; $d.S := 0$;

{ Invariant : $\langle \forall u : u \in b : \langle S| \cdot D^* \cdot |u\rangle = d.u \rangle$

$\wedge \langle \forall u : u \notin b : \langle S| \cdot D^* \cdot |u\rangle = \langle \Downarrow v : v \in g : d.v + \langle v| \cdot D^* \cdot |u\rangle \rangle\rangle$

}

do

$F \notin b \rightarrow$ **if** $\langle \Box v : v \in g \wedge d.v + H.v = \langle \Downarrow u : u \in g : d.u + H.u \rangle$

$: b := b \cup \{v\}$; $g := g - \{v\}$;

for each edge (v,w) from v **do**

$d.w := d.w \downarrow (d.v + \langle v| \cdot D \cdot |w\rangle)$;

$g := g \cup (\{w\} - (b \cup g))$

rof

$\rangle$

fi

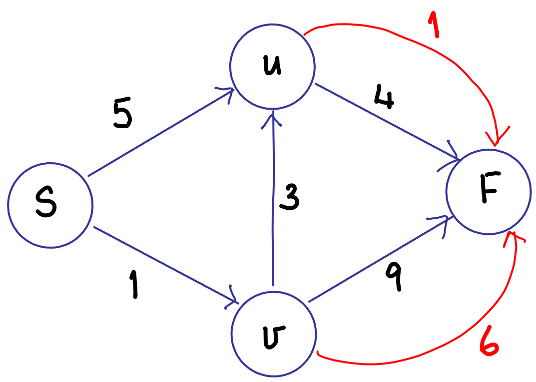
od

{ $\langle S| \cdot D^* \cdot |F\rangle = d.F$ }

Inadmissible Heuristic

"heuristic" distance

actual distance



$$\langle S| \cdot D^* \cdot |F\rangle \quad = \quad 8$$

Verification. 1. $\quad b,g \;:=\; b \cup \{v\},\, g - \{v\}$

$$
\begin{array}{ll}
 & d.v \;=\; \langle S| \cdot D^* \cdot |v\rangle \\
= & \quad \{ \text{ Invariant}, \quad v \notin b \; \} \\
 & d.v \;=\; \langle \Downarrow w : w \in g : d.w + \langle w| \cdot D^* \cdot |v\rangle \rangle \\
= & \quad \{ \text{ arithmetic, addition distributes over } \downarrow \; \} \\
 & d.v + H.v \;=\; \langle \Downarrow w : w \in g : d.w + \langle w| \cdot D^* \cdot |v\rangle + H.v \rangle \\
= & \quad \{ \; v \in g, \text{ range splitting } \; \} \\
 & d.v + H.v \;\leq\; \langle \Downarrow w : w \in g \wedge w \neq v : d.w + \langle w| \cdot D^* \cdot |v\rangle + H.v \rangle \\
\Leftarrow & \quad \{ \; H \text{ is admissible. I.e. } H \leq D \cdot H \,. \\
 & \qquad \text{Hence} \quad H \leq D^+ \cdot H \,. \\
 & \qquad [\, w \neq v \;\Rightarrow\; \langle w| \cdot D^* \cdot |v\rangle \;=\; \langle w| \cdot D^+ \cdot |v\rangle \,] \;\; \} \\
 & d.v + H.v \;\leq\; \langle \Downarrow w : w \in g \wedge w \neq v : d.w + H.w \rangle \\
= & \quad \{ \text{ range splitting}, \quad v \in g \; \} \\
 & d.v + H.v \;=\; \langle \Downarrow w : w \in g : d.w + H.w \rangle
\end{array}
$$

Exercise: modify A* algorithm in same way that the wave algorithm was modified

— assume same heuristic H but choose edges rather than nodes.

Can the same implementation technique be used for the grey edges?