Planning as heuristic search

Blai Bonet, Héctor Geffner

Presented by Lukas Probst

Focus of the paper

- General problem solvers, which are able to solve different domains with the same code
  - Introduce two heuristics based on delete relaxation
  - Present two different heuristic search planers

➡ Focus on non-optimal sequential planning

Structure

- State model
- Strips state model
- Delete relaxation
  - Optimal relaxation heuristic
  - Approximations
    - h_{add}
    - h_{max}
- Heuristic search planners
- Evaluation

State model (formal)

A state model is a tupel $\mathcal{S} = \langle S, s_0, S_G, A, f, c \rangle$, where

- S is a finite and non-empty set of states s,
- $s_0 \in S$ is the initial state,
- $S_G \subseteq S$, is a non-empty set of goal states,
- $A(s) \subseteq A$ denotes the actions applicable in each state $s \in S$,
- $f(a,s)$ denotes a state transition function for all $s \in S$ and $a \in A(s)$,
- $c(a,s)$ stands for the cost of doing action a in state s

State model (formal)

A **solution** of a state model is a sequence of actions $a_0, a_1, \ldots, a_n$ that generates a state trajectory $s_0, s_1 = f(s_0), \ldots, s_{n-1} = f(a_n, s_n)$ such that each action $a_i \in \mathrm{A}(s_i)$ and $s_{n+1} \in S_G$.

The solution is **optimal** when the total cost $\sum_{i=0}^{n} c(a_i, s_i)$ is minimized.

Strips state model (formal)

A planning problem is represented by a tuple $P = \langle A, O, I, G \rangle$ where

- A is a set of atoms,
- O is a set of operators,
- $I \subseteq A$ encodes the initial situation and
- $G \subseteq A$ encodes the goal situation.

Each operator $op \in O$ has

- a precondition list $Prec(op) \subseteq A$,
- a add list $Add(op) \subseteq A$ and
- a delete list $Del(op) \subseteq A$.

Strips state model (formal)

A Strips problem $P = \langle A, O, I, G \rangle$ defines a state space

$\mathcal{S}_{\mathcal{P}} = \langle S, s_0, S_G, A, f, c \rangle$ where

- the states $s \in S$ are collections of atoms from A,
- the initial state s_0 is I,
- the goal states $s \in S_G$ are such that $G \subseteq s$,
- the actions $a \in A(s)$ are the operators $op \in O$ such that $Prec(op) \subseteq s$,
- the transition function f maps states s into states $s' = s - Del(a) + Add(a)$ for $a \in A(s)$ and
- all action costs $c(a,s)$ are 1.

Solve a Strips problem

- (Optimal) solutions of the problem P are the (optimal) solutions of the state model $\mathcal{S}_{\mathcal{P}}$.

⇨ Perform a search in that search space.

- For the search in the search space we need a heuristic.

Delete relaxation

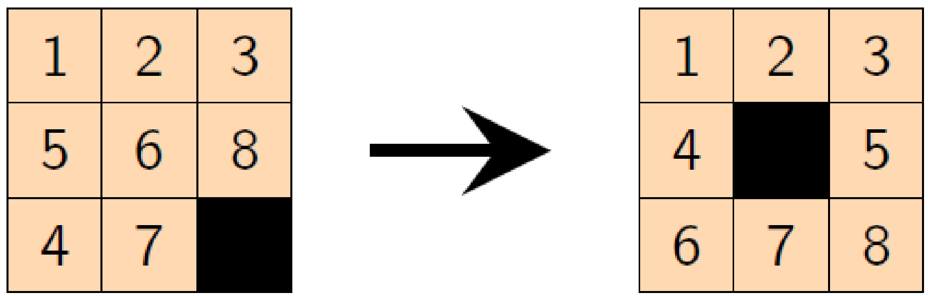
- Idea: Ignore all delete effects because they are always harmful.

➡ The relaxed Problem P^+ is equivalent to the Problem P but $Del(op) = \varnothing \quad \forall op \in O.$

- For any state s, the optimal cost $h^+(s)$ for solving the relaxed problem P^+ can be shown to be a lower bound on the optimal cost $h^*(s)$ for solving the original problem P.

Optimal relaxation heuristic

- $h^{+}(s)$ could be used as an admissible heuristic for solving the original problem P.



- $h^{MD}(s) = 6 \quad < \quad h^{+}(s) = 7 \quad < \quad h^{*}(s) = 8$

- Unfortunately also solve P^{+} is NP-hard.

➡ We need an approximation for $h^{+}(s)$

Approximation (formal)

- Cost of achieving an atom *p* from the state *s* :

$$g_s(p) = \begin{cases} 0 & \text{if } p \in s \\ \min\limits_{op \in O(p)} \left[1 + g_s(\mathrm{Pr}ec(op))\right] & \text{otherwise} \end{cases}$$

  - where *O(p)* stands for the actions *op* that add *p*
  - and $g_s(Prec(op))$ stands for the estimated cost of achieving the preconditions of action *op* from *s*.

Approximation (formal)

- The expression $g_s(Prec(op))$ stands for the estimated cost of the set of atoms given by *Prec(op)*

- The resulting heuristic *h(s)* that estimates the cost of achieving the goal *G* from a state *s* is defined as:

$$h(s) \overset{\text{def}}{=} g_s(G)$$

Approximation (formal)

- Problem: How to calculate $g_s(C)$, if C is a set of atoms?

- This can be done in different ways!

- Two ways are presented in this paper:
  - h_{add}
  - h_{max}

Additive heuristic (h_{add})

- Additive costs: Sum of the costs of the individual atoms in *C*

$$h_{add}(G) = g_s^+(G) = \sum_{r \in G} g_s(r)$$

- Assumes that subgoals are independent
  - **pessimistic**: assumes all atoms have to be reached independently
  - this is not true in general
  - h_{add} may overestimate the costs

⇨ h_{add} is **not admissible**!

Max heuristic (h_{max})

- Max costs: Maximal cost for reaching a single atom $r \in G$

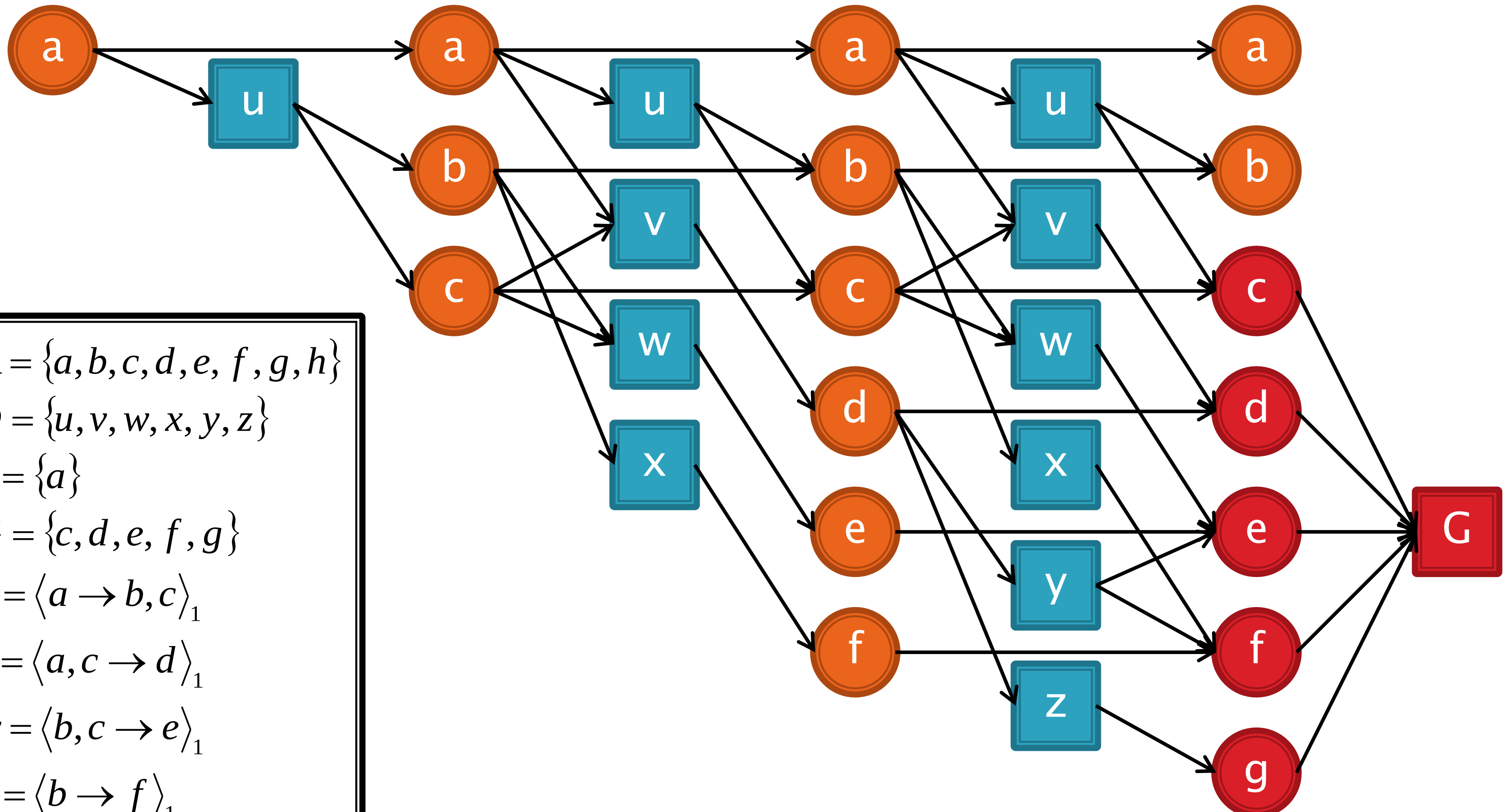
$$h_{\max}(G) = g_s^{\max}(G) = \max_{r \in G} g_s(r)$$

- **Optimistic**: assumes that by reaching the most difficult subgoal (atom), all other subgoals are reached too.

➡ h_{max} is admissible.

- But: h_{max} is often **less informative** than h_{add}

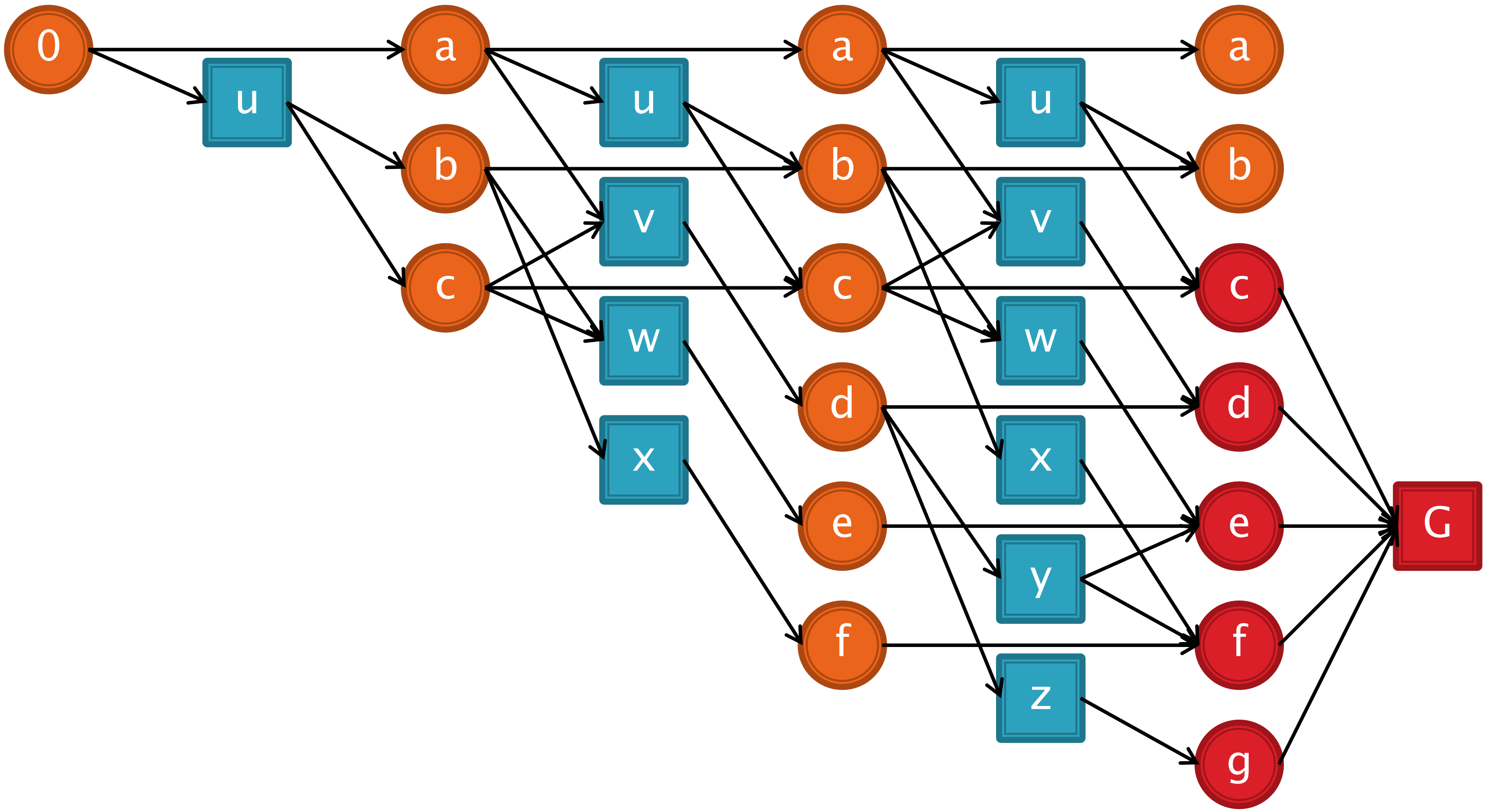
Example relaxed planning graph



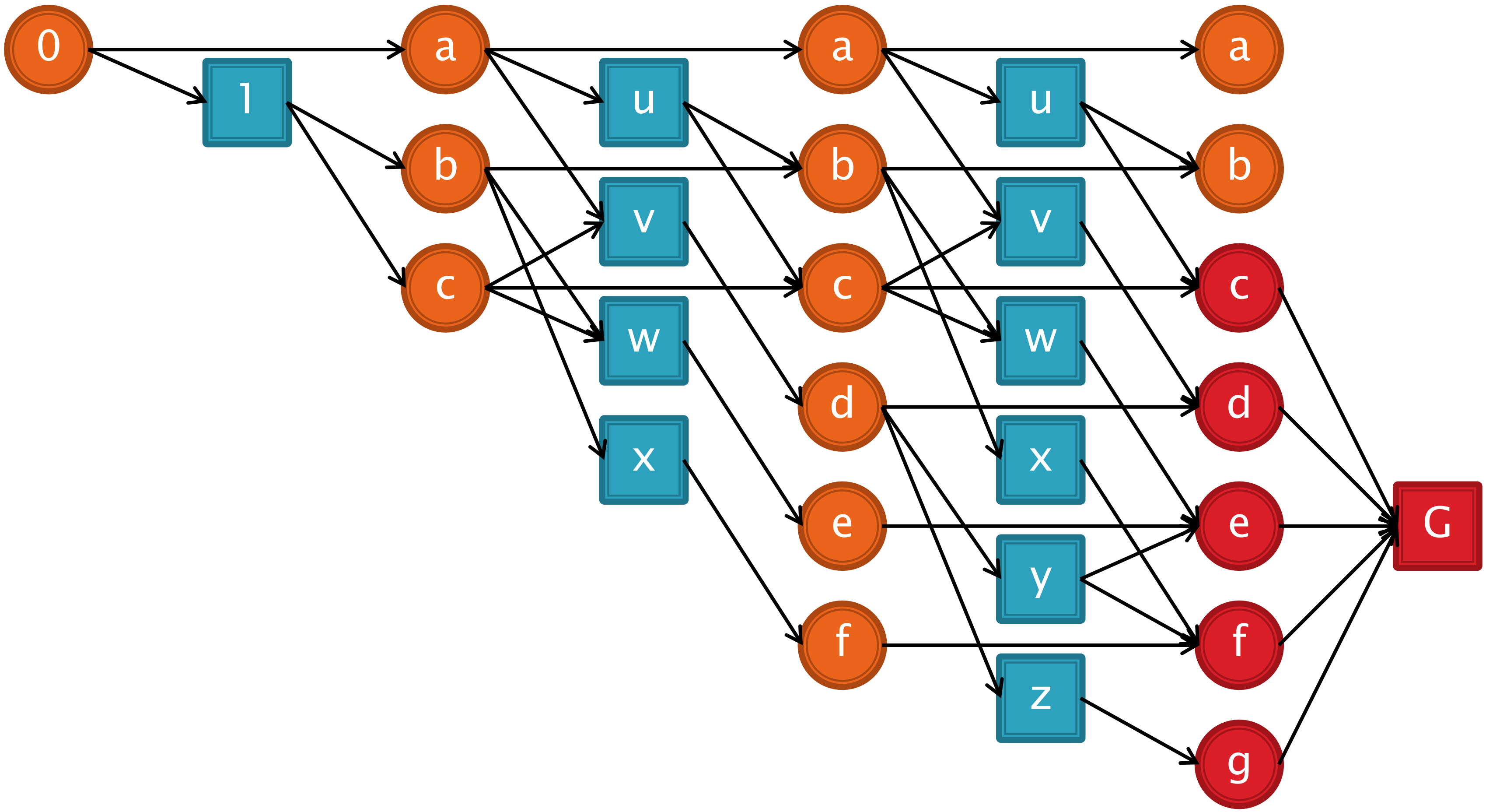
$A = \{a, b, c, d, e, f, g, h\}$
$O = \{u, v, w, x, y, z\}$
$I = \{a\}$
$G = \{c, d, e, f, g\}$
$u = \langle a \to b, c \rangle_1$
$v = \langle a, c \to d \rangle_1$
$w = \langle b, c \to e \rangle_1$
$x = \langle b \to f \rangle_1$
$y = \langle d \to e, f \rangle_1$
$z = \langle d \to g \rangle_1$

$h^*(\{a\}) = 4$

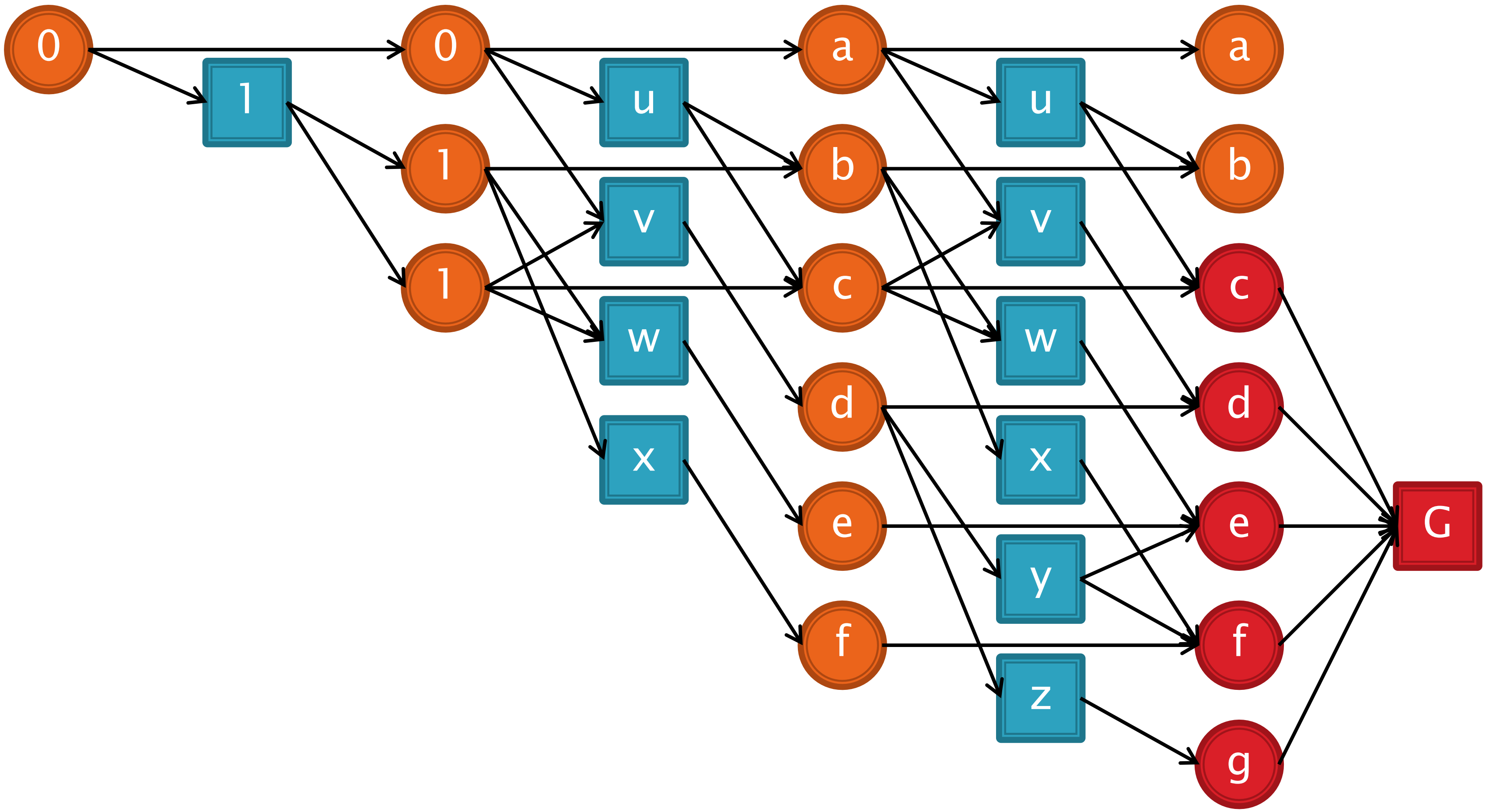
h_{add} example



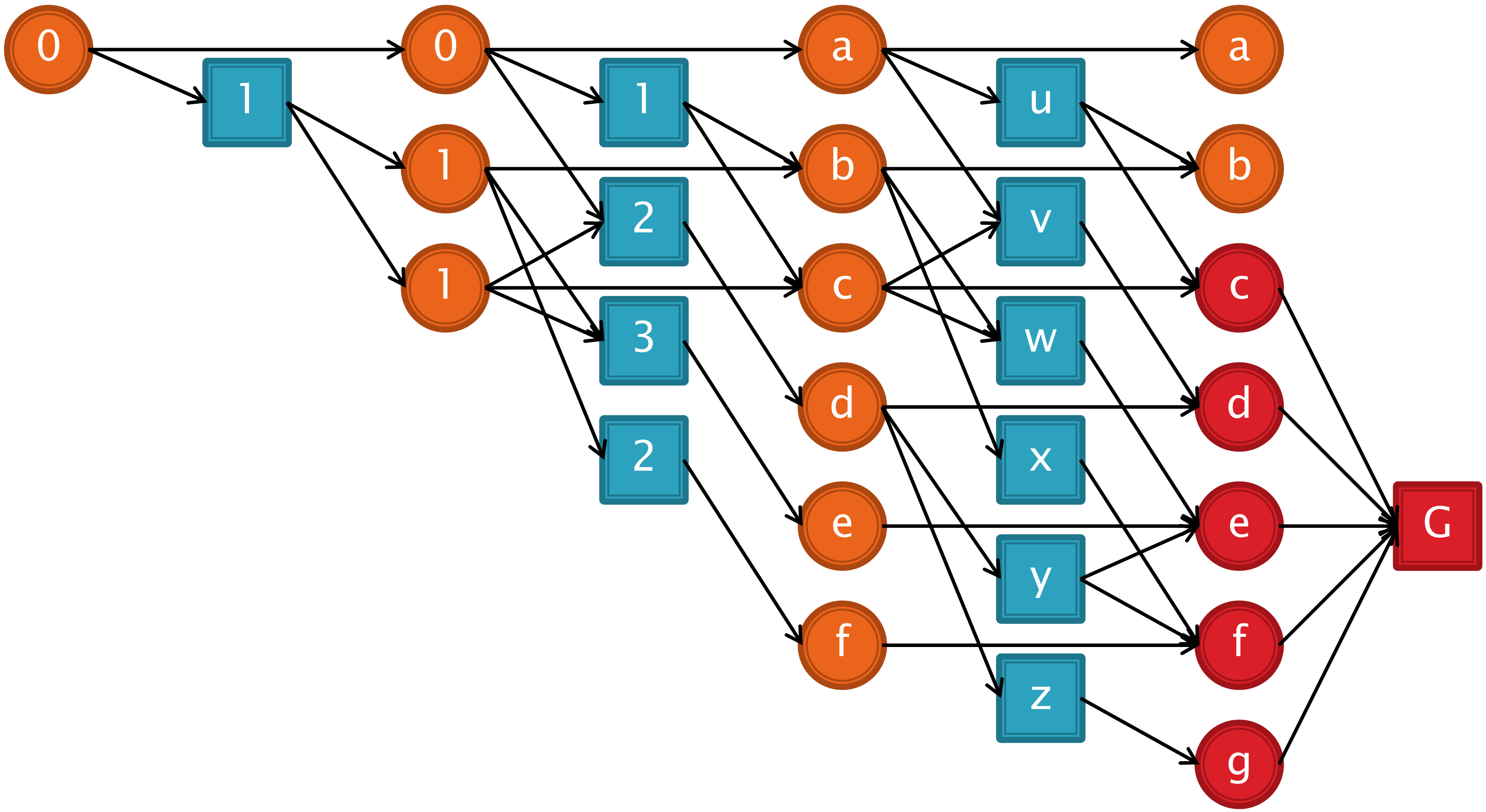
h_{add} example



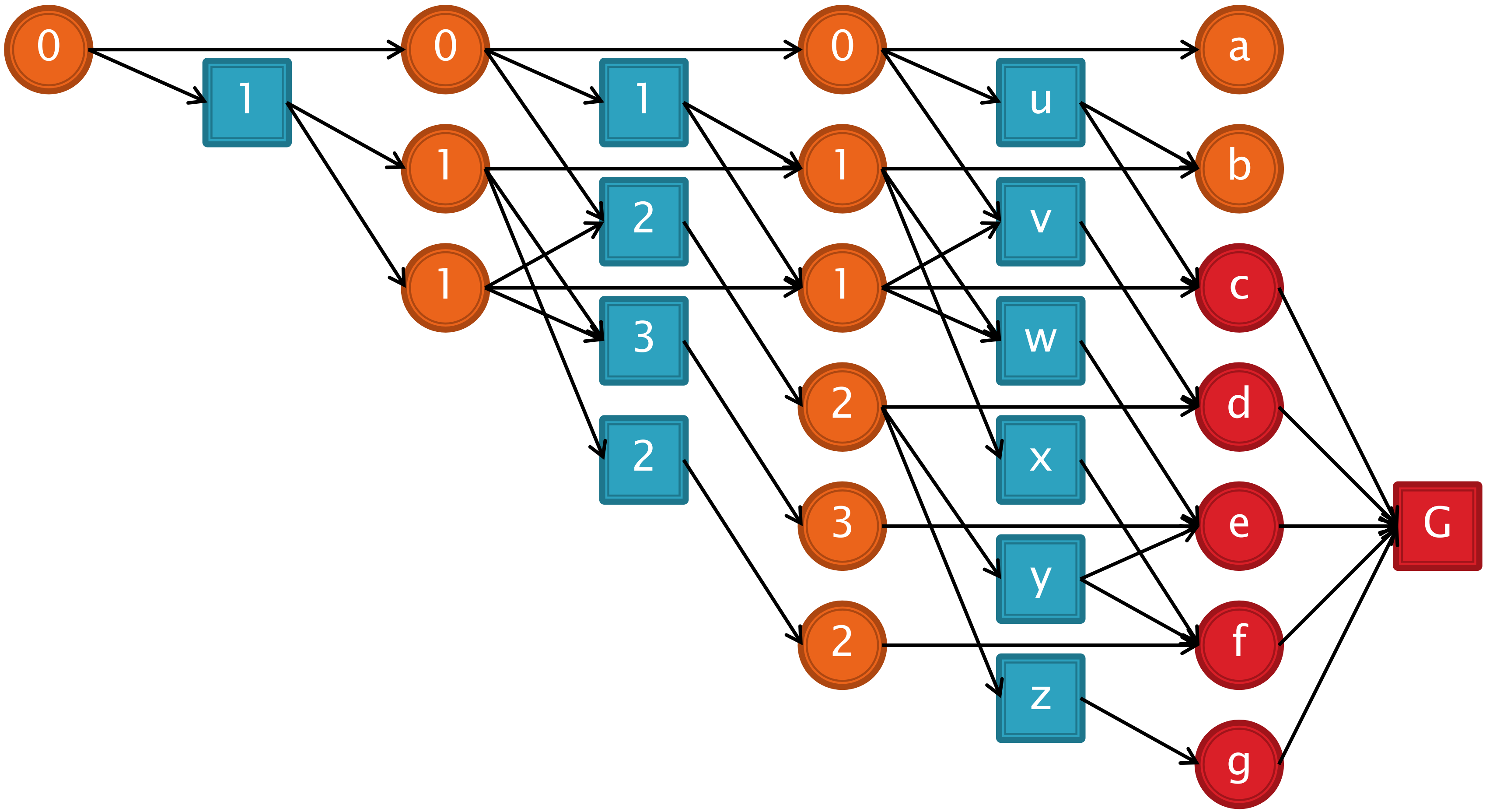
h_{add} example



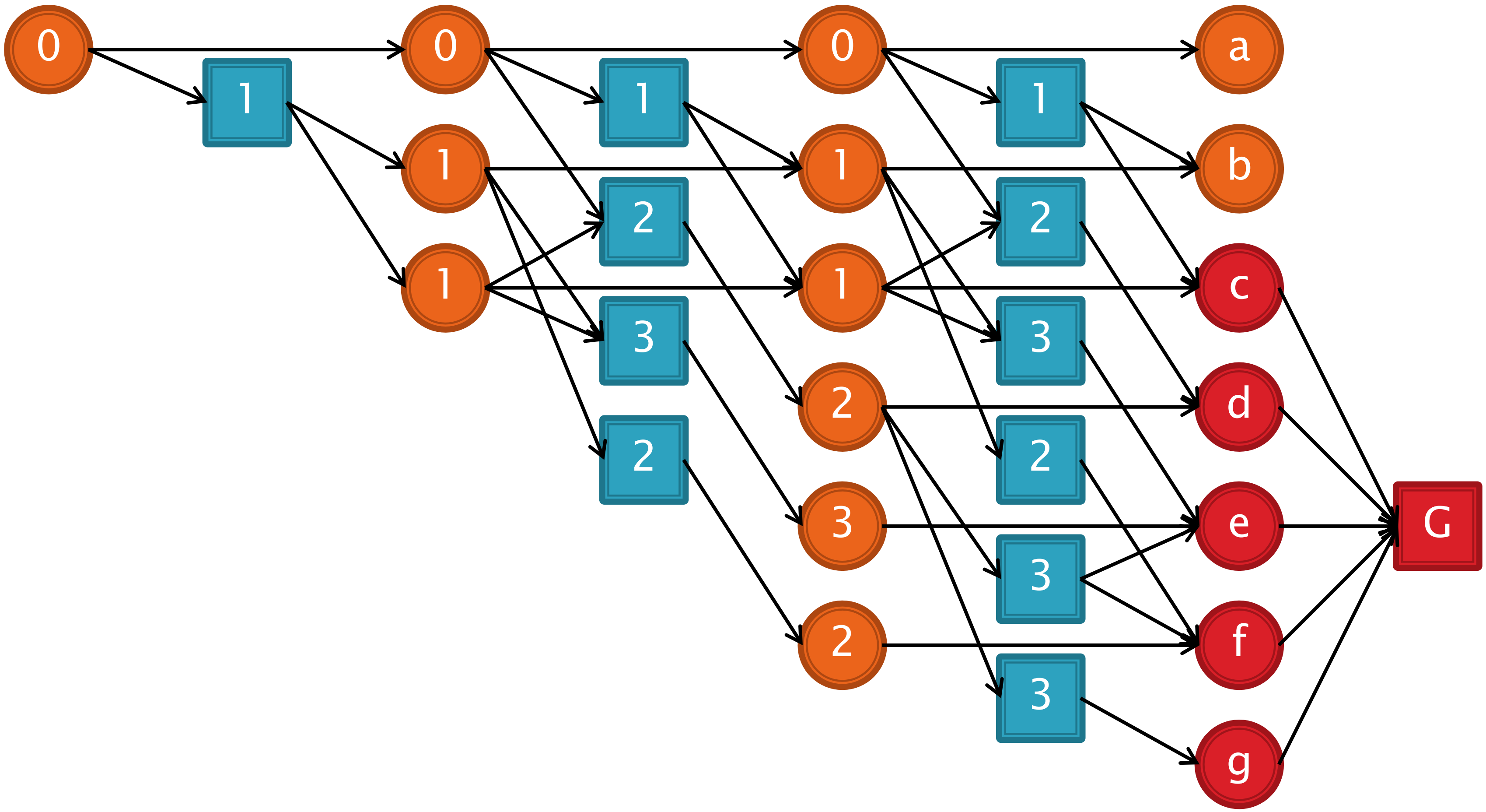
h_{add} example



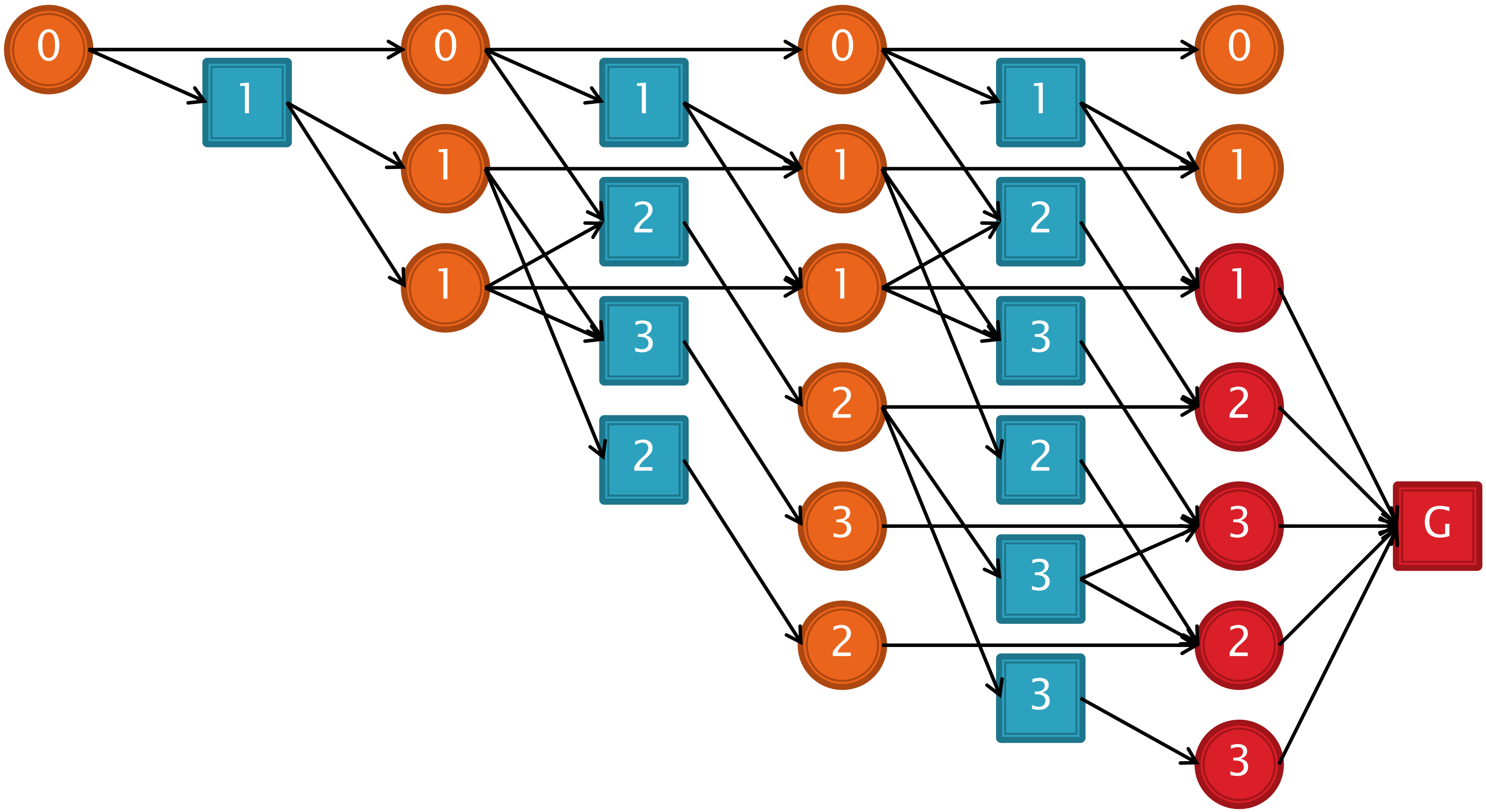
h_{add} example



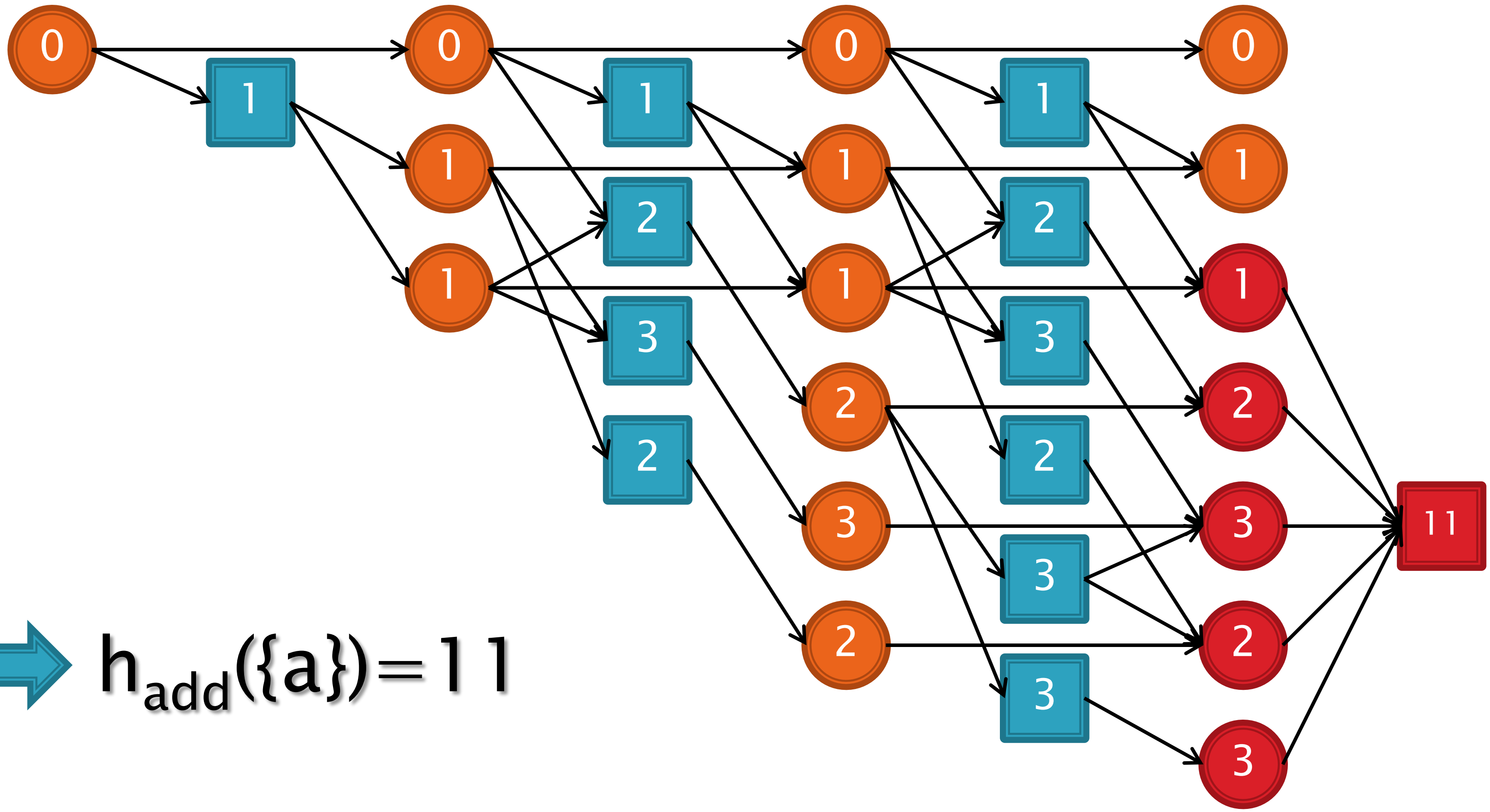
h_{add} example



h_{add} example

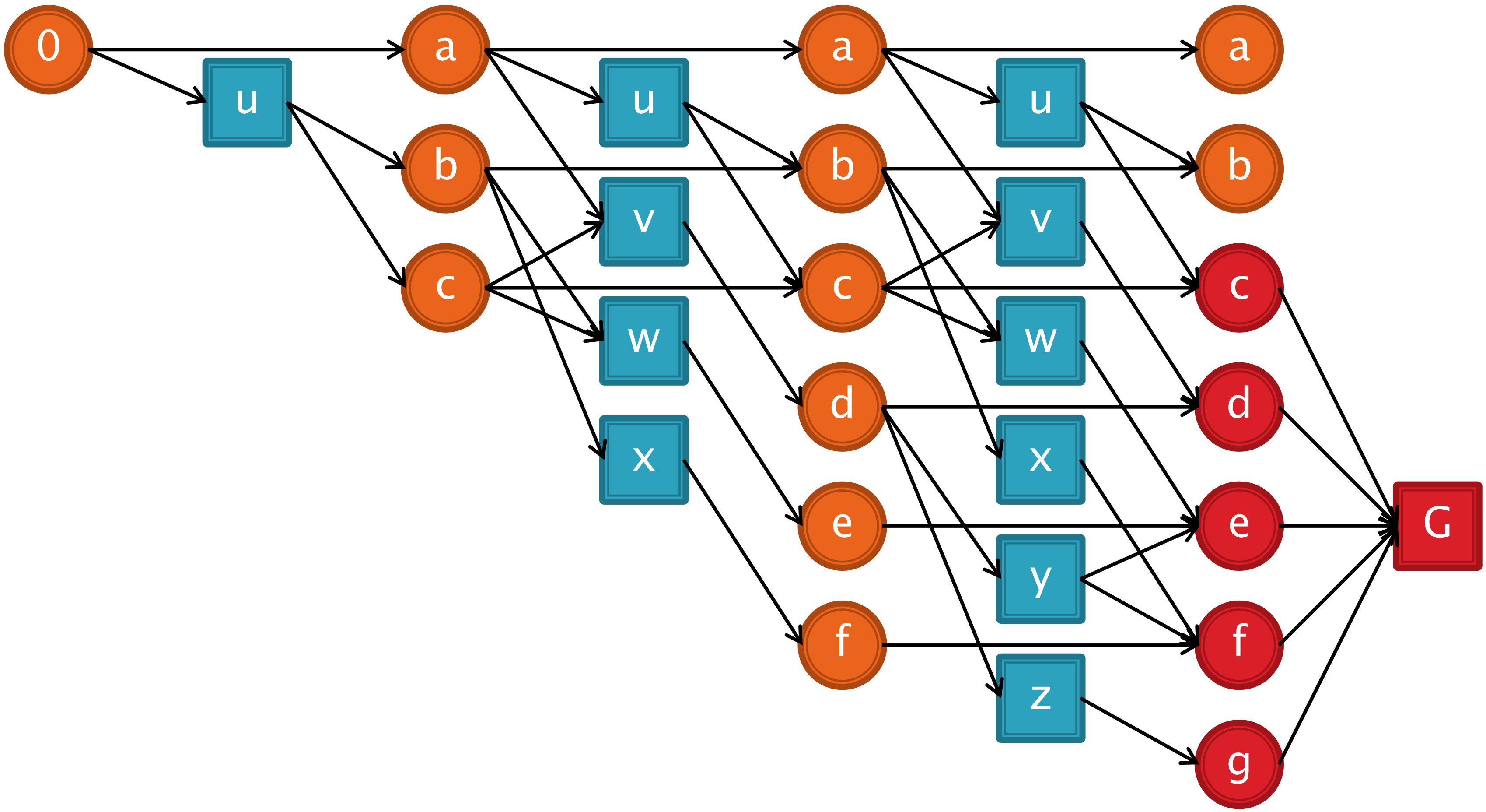


h_{add} example

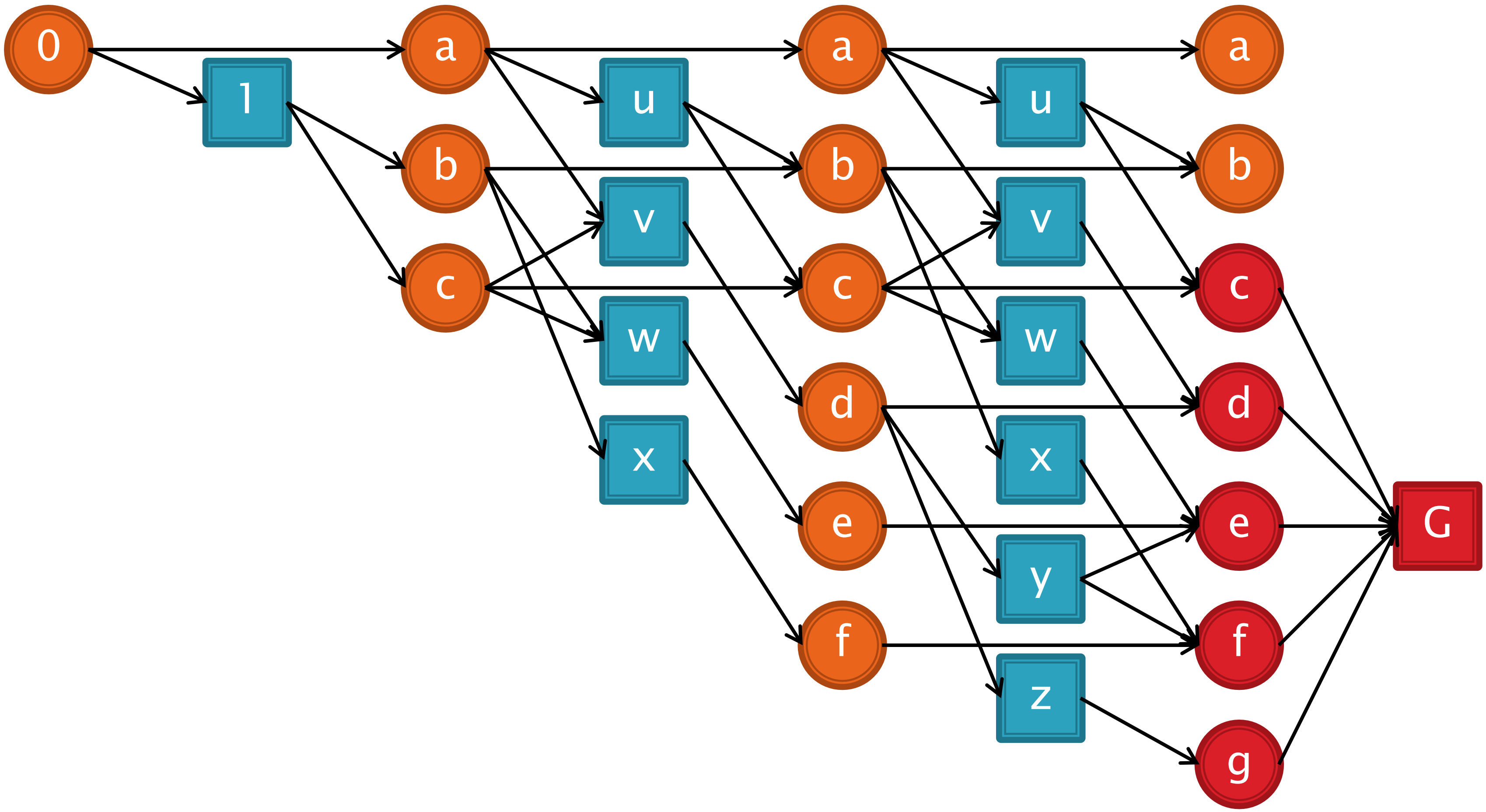


$\Rightarrow$ $h_{add}(\{a\})=11$

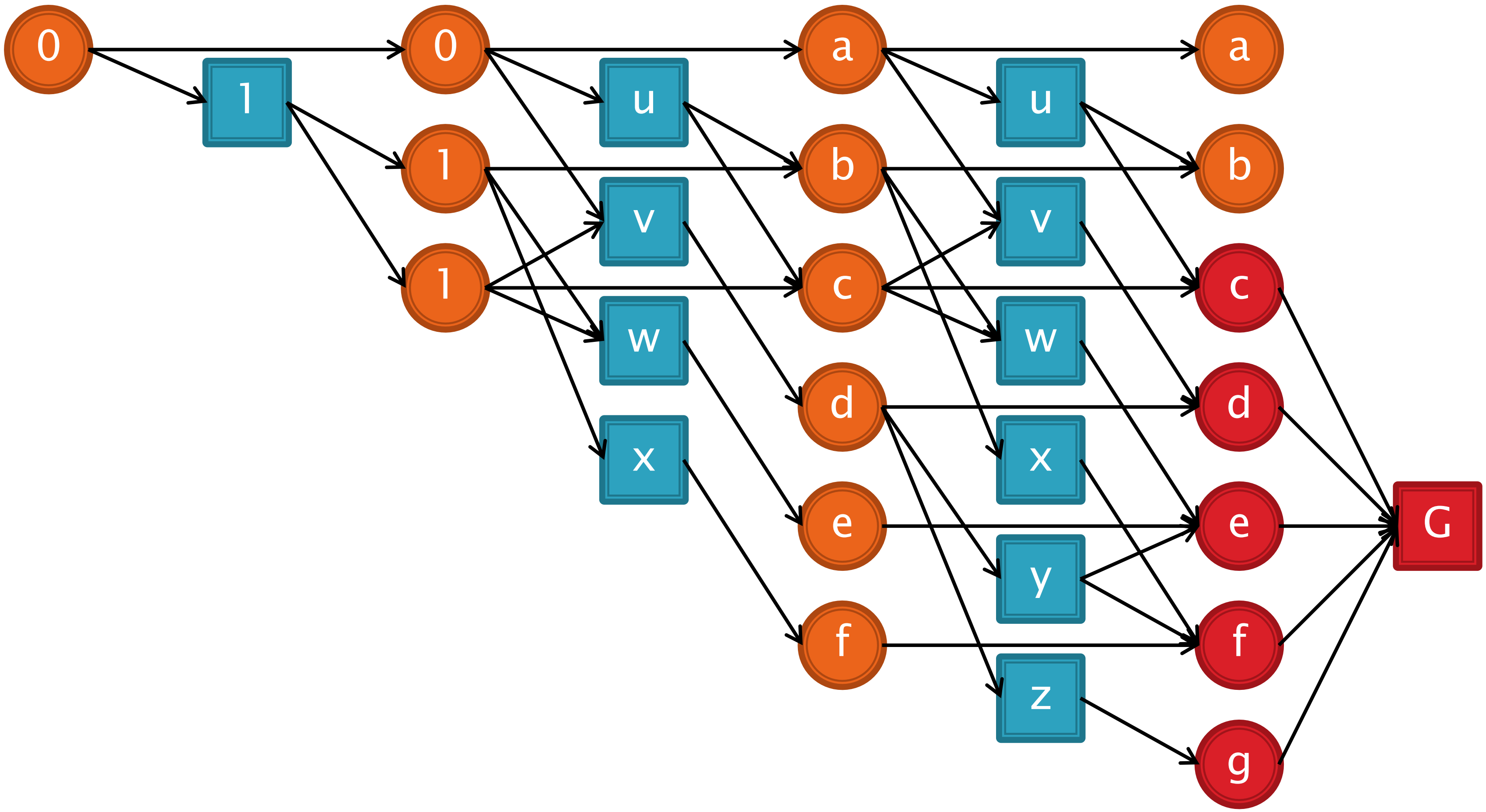
h_{max} example



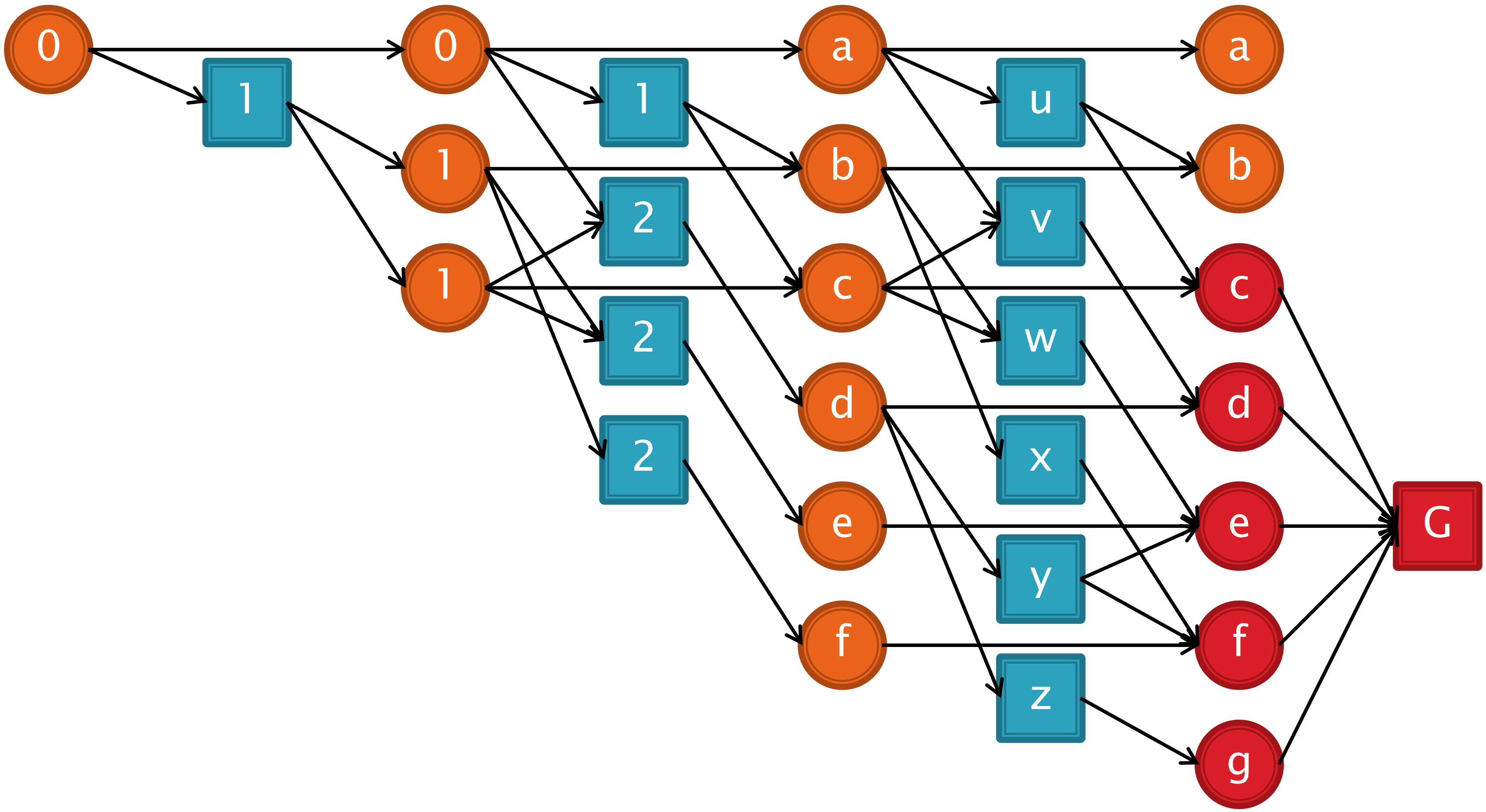
h_{max} example



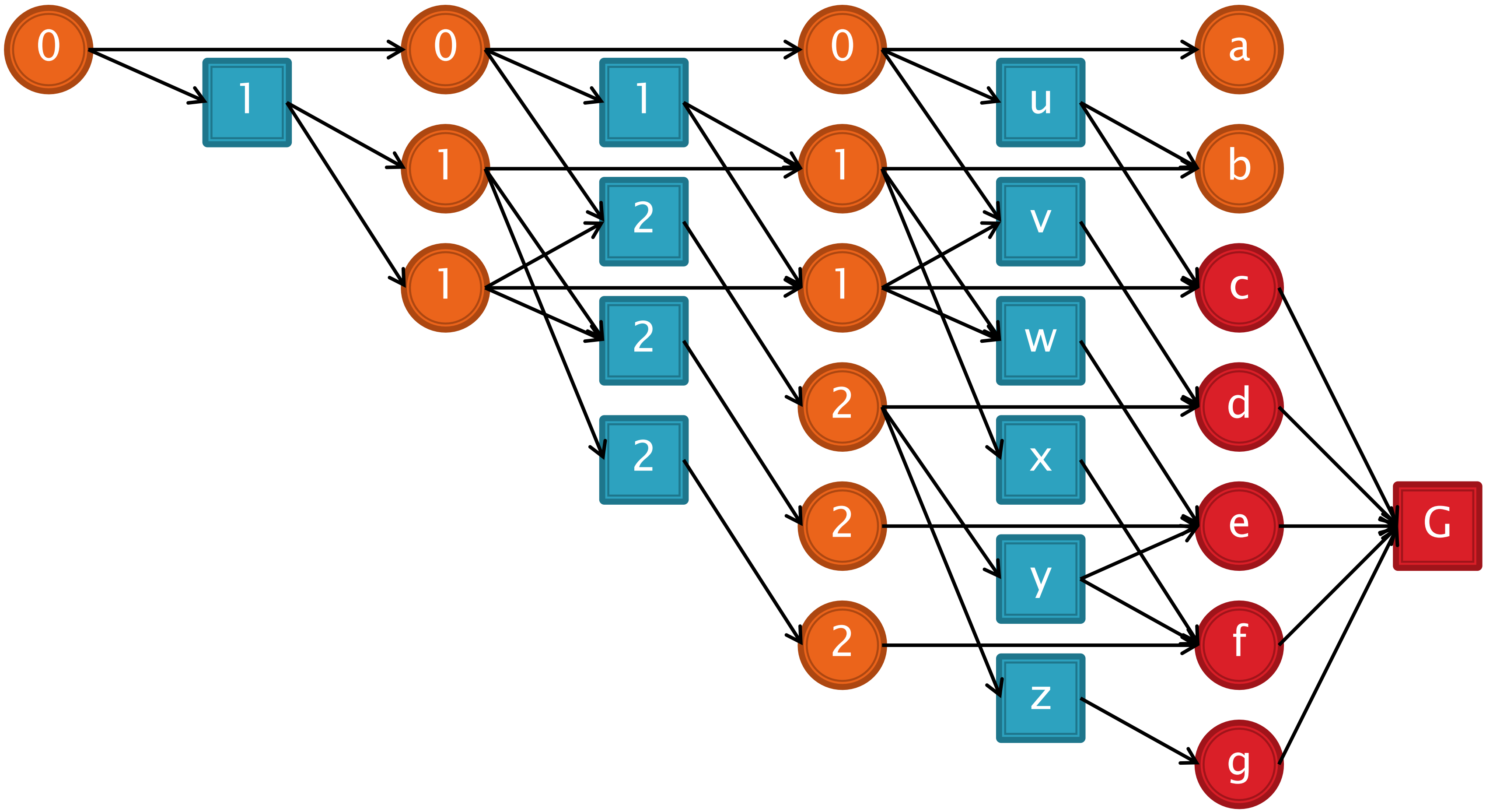
h_{max} example



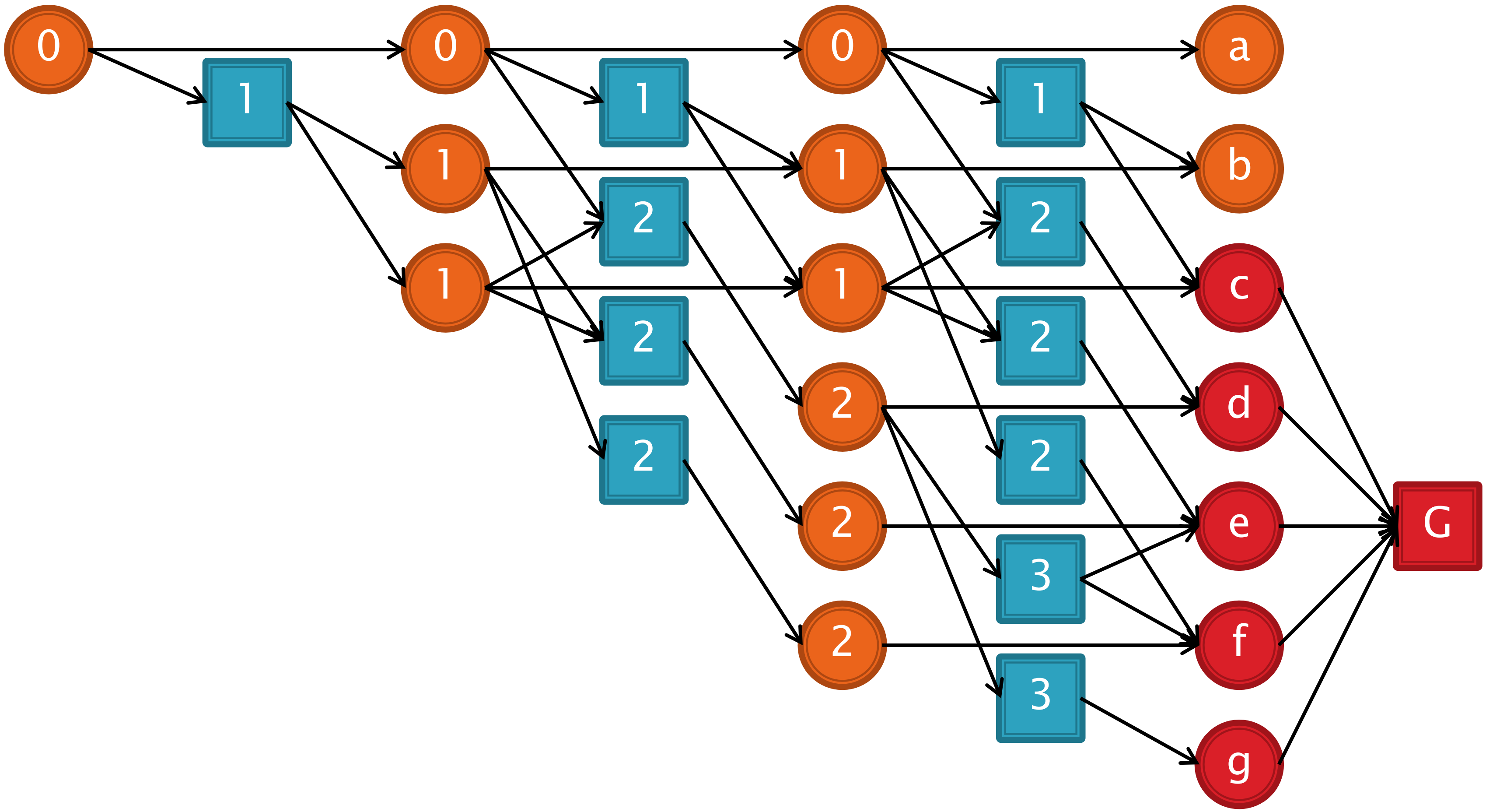
h_{max} example



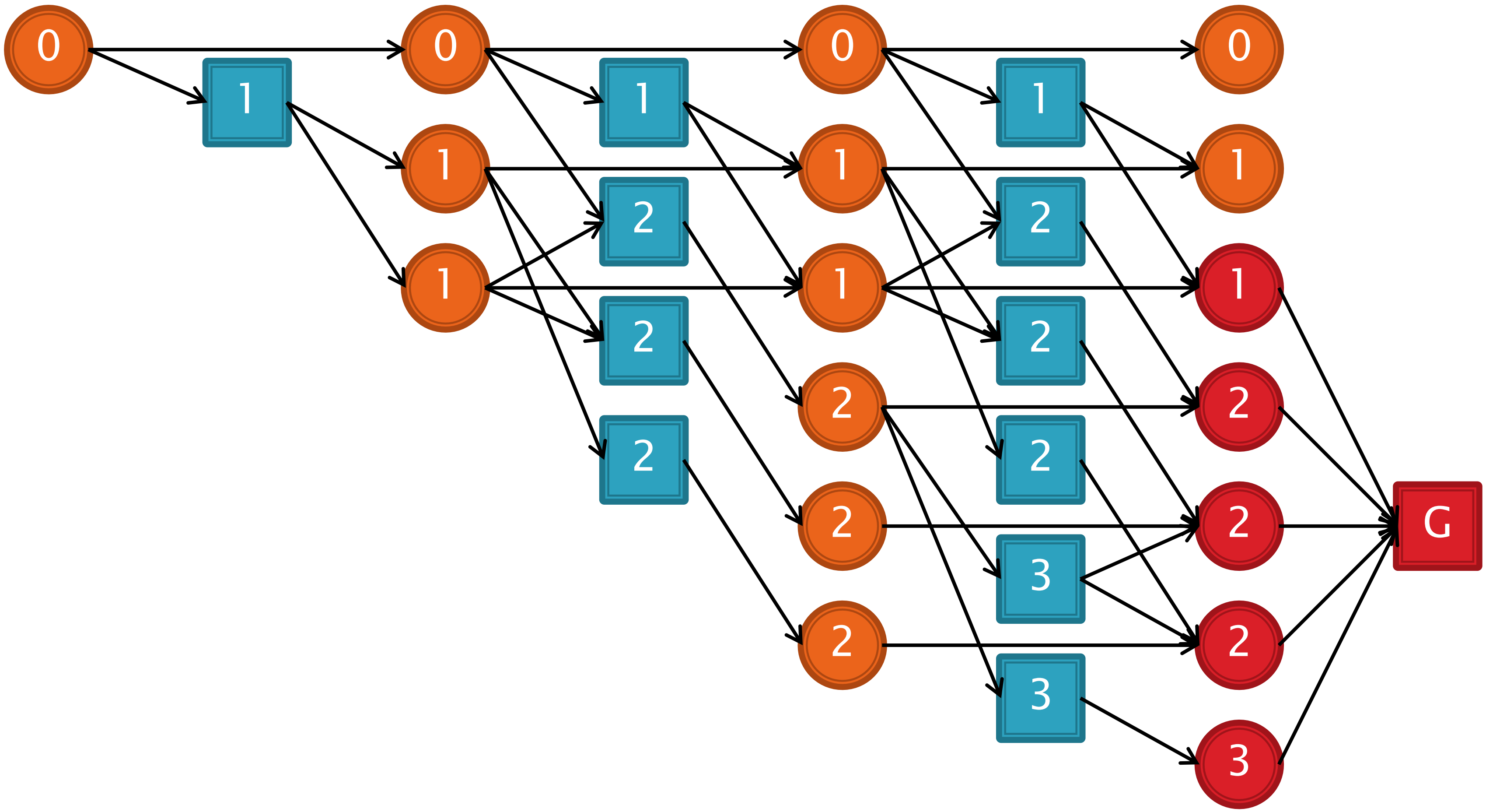
h_{max} example



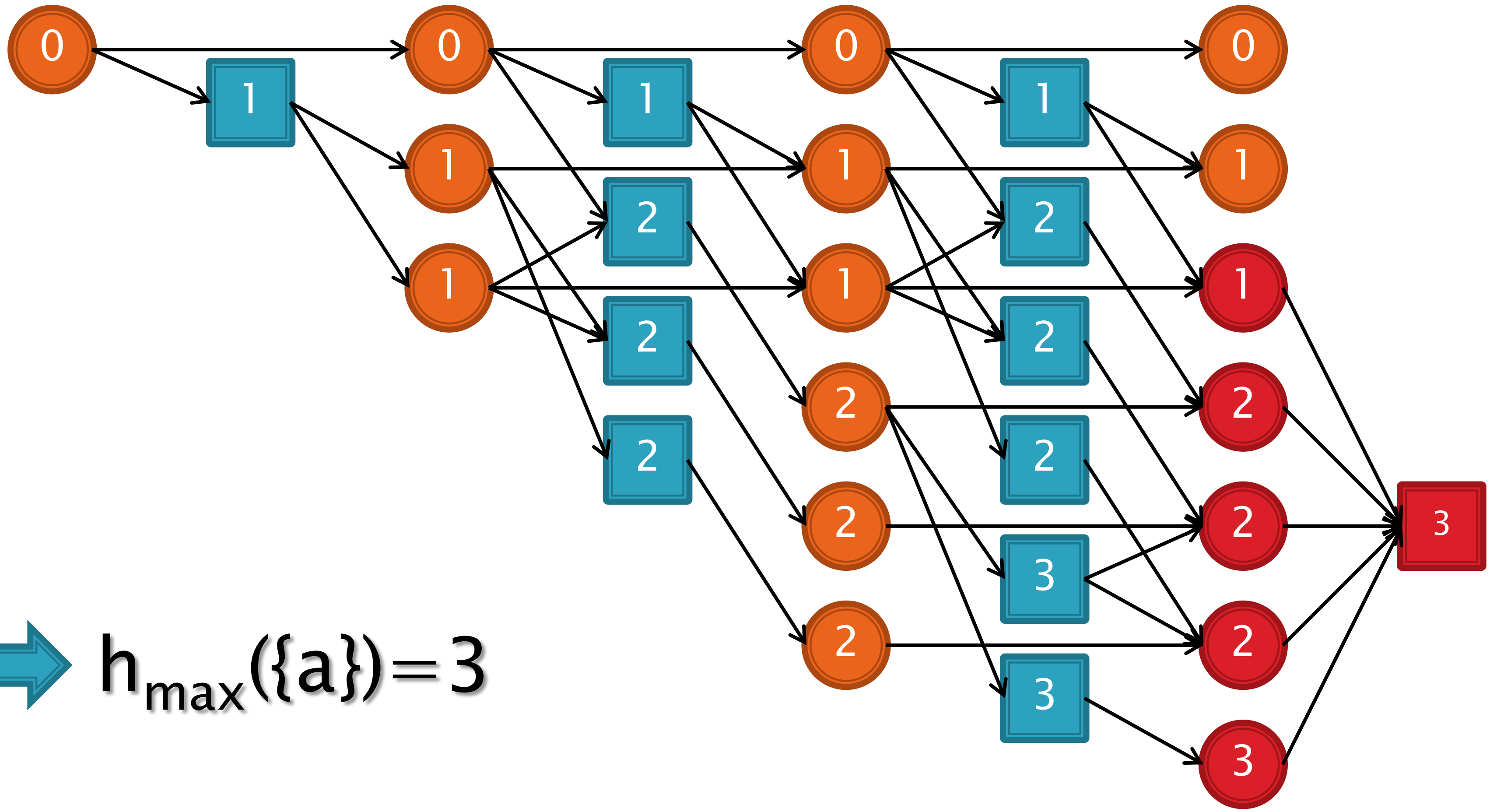
h_{max} example



h_{max} example



h_{max} example



$\Rightarrow h_{max}(\{a\})=3$

Heuristic search planners

- Two different heuristic search planners:
  - HSP: A hill-climbing planner
  - HSP2: A best-first search planner
- Use h_{add} heuristic
- Both find non-optimal solutions

Evaluation focus

- Compare:
  - How good are the found solutions? (steps)
  - How long did the algorithm need to find the solution? (time)
- Also compare the heuristic search planners to the 3 best current optimal parallel planners:
  - IPP
  - STAN
  - BLACKBOX

HSP: A hill-climbing planner

- Hill-climbing search:
  - At every step, one of the best children (minimize h_{add}) is selected for expansion.
  - This process is repeated until the goal is reaches.
  - Ties are broken randomly.

➡ Estimated atom costs $g_s(p)$ and the heuristic $h(s)$ are computed for all states s that are created.

- Not complete
- Extensions for plateaus and already visited states.

Evaluation: HSP vs. BLACKBOX vs. IPP vs. STAN

- HSP solved more problems than the other planners but it often took more time or produced longer plans

Round	Planner	Avg. time	Solved	Fastest	Shortest
Round 1	BLACKBOX	1.49	63	16	55
	HSP	35.48	82	19	61
	IPP	7.40	63	29	49
	STAN	55.41	64	24	47
Round 2	BLACKBOX	2.46	8	3	6
	HSP	25.87	9	1	5
	IPP	17.37	11	3	8
	STAN	1.33	7	5	4

HSP2: A best-first search planner

- Best-first search from the initial state to the goal
- Weighted A*: $f(n) = g(n) + W \cdot h(n)$
  - Higher values of *W* usually lead to the goal faster but with solutions of lower quality.
  - If the heuristic admissible, the solutions found by WA* are guaranteed not to exceed the optimal costs by more than a factor of *W*.
    (BUT: h_{add} is non-admissible)
  - *W=5* for the evaluation

Evaluation: HSP vs. HSP2 vs. STAN vs. BLACKBOX

- Considered domains: Blocks, Logistics, Gripper, 8-Puzzle, Hanoi and Tire-World
- Results:
  - HSP and HSP2 are capable of solving the problems solved by the two state-of-the-art planners (STAN, BLACKBOX)
  - In some domains, HSP and in particular HSP2 solve problems that the other planners with their default settings do not currently solve.
  - HSP2 tends to be faster and more robust than HSP.

Evaluation: Choice of W in HSP2

- Typical (e.g., Hanoi): values in the interval *[2,10]* produce similar results.
  - h_{add} is not admissible → overestimates the true costs without the need of a multiplying factor
- Logistics & Gripper: $W=1$ does not lead to solutions
  - involve subgoals that are mostly independent → h_{add} is not sufficiently overestimating
- Sliding-Puzzle: values closer to 1 produce better solutions in more time
  - Correspondence with the normal pattern observed in cases in which the heuristic is admissible
  - Works because branching factor is small

Questions?