

Competitive Online Transportation Simplified

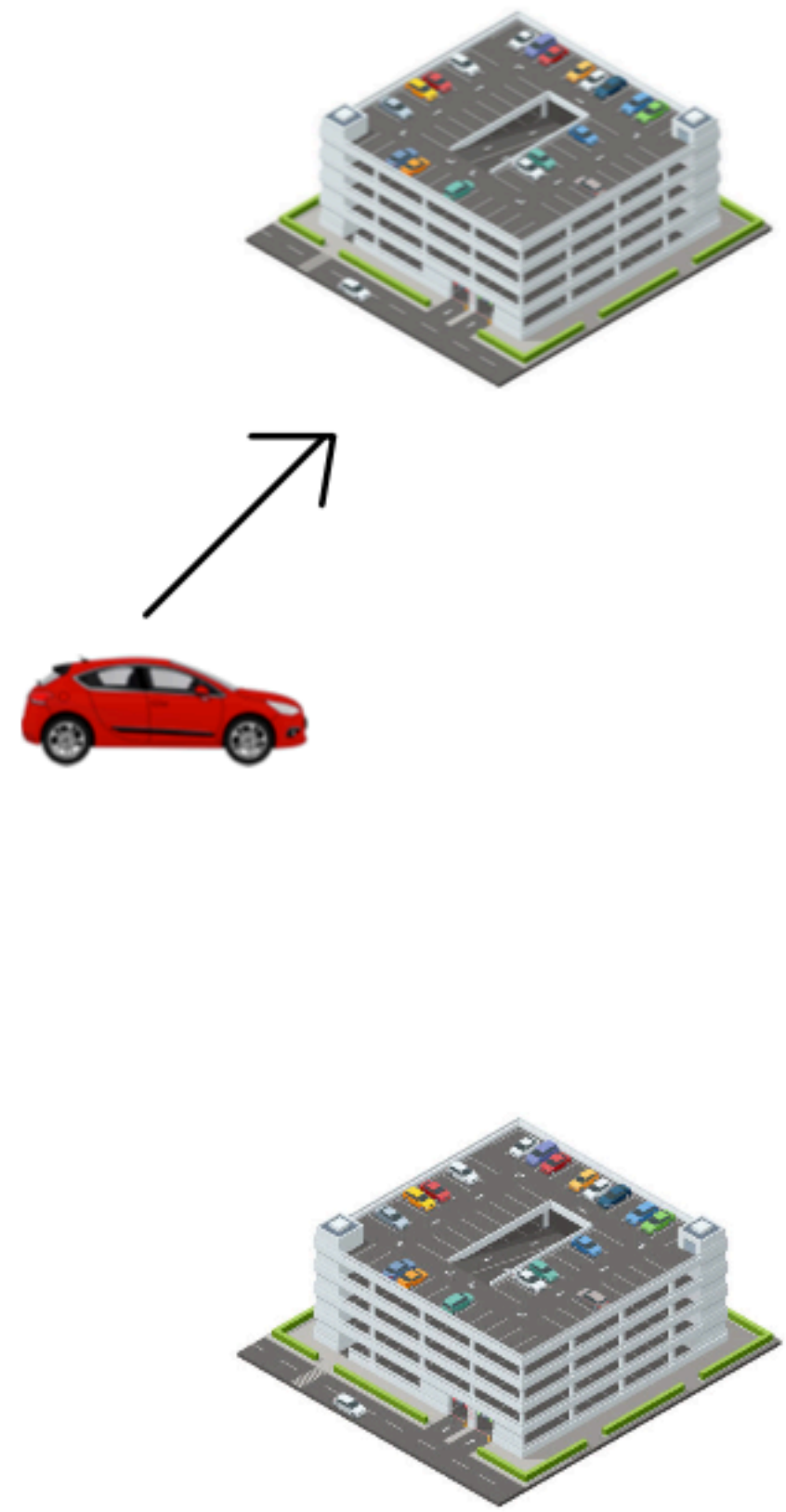
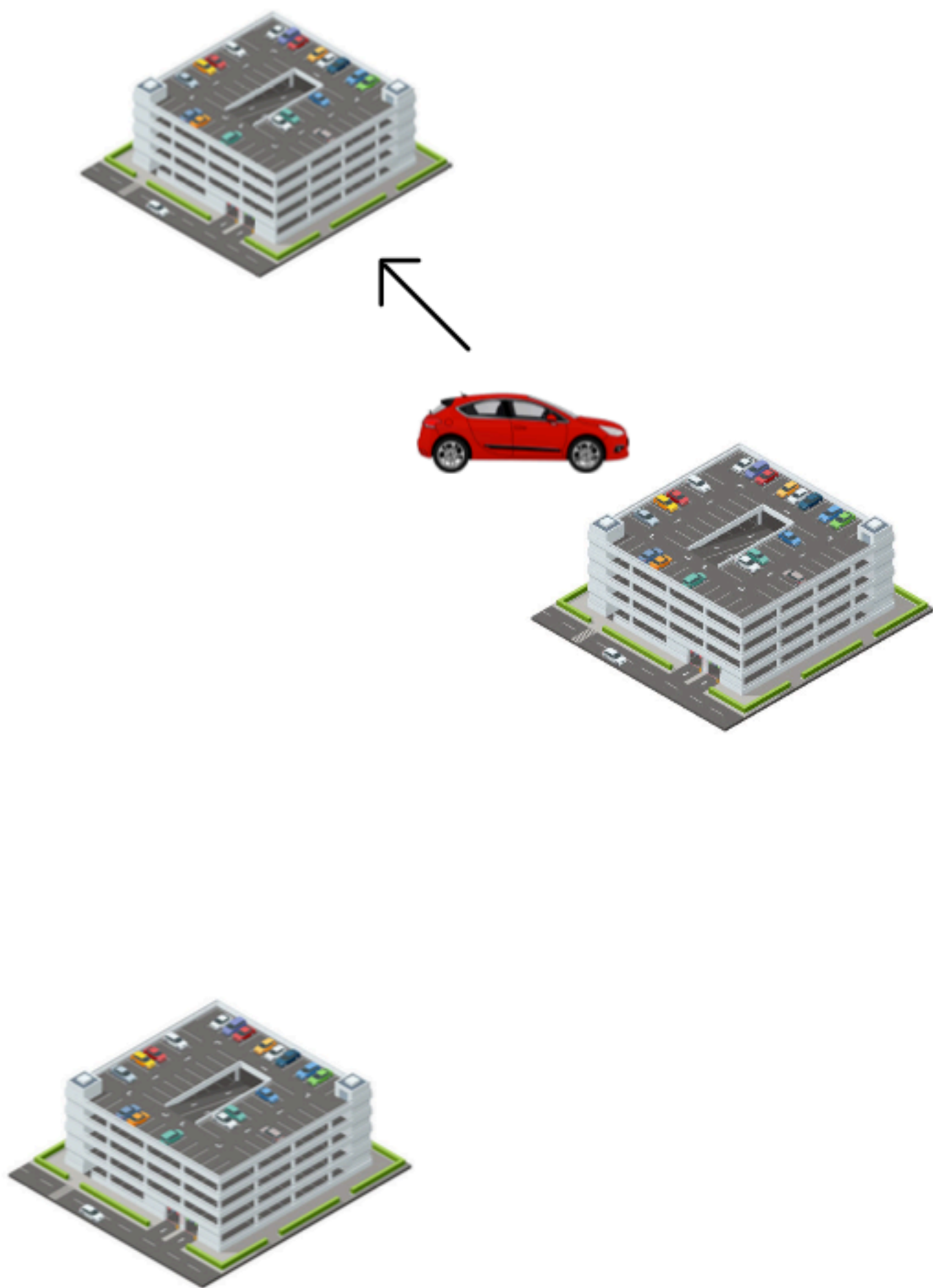
Stephen Arndt, Benjamin Moseley, Kirk Pruhs, and Marc Uetz



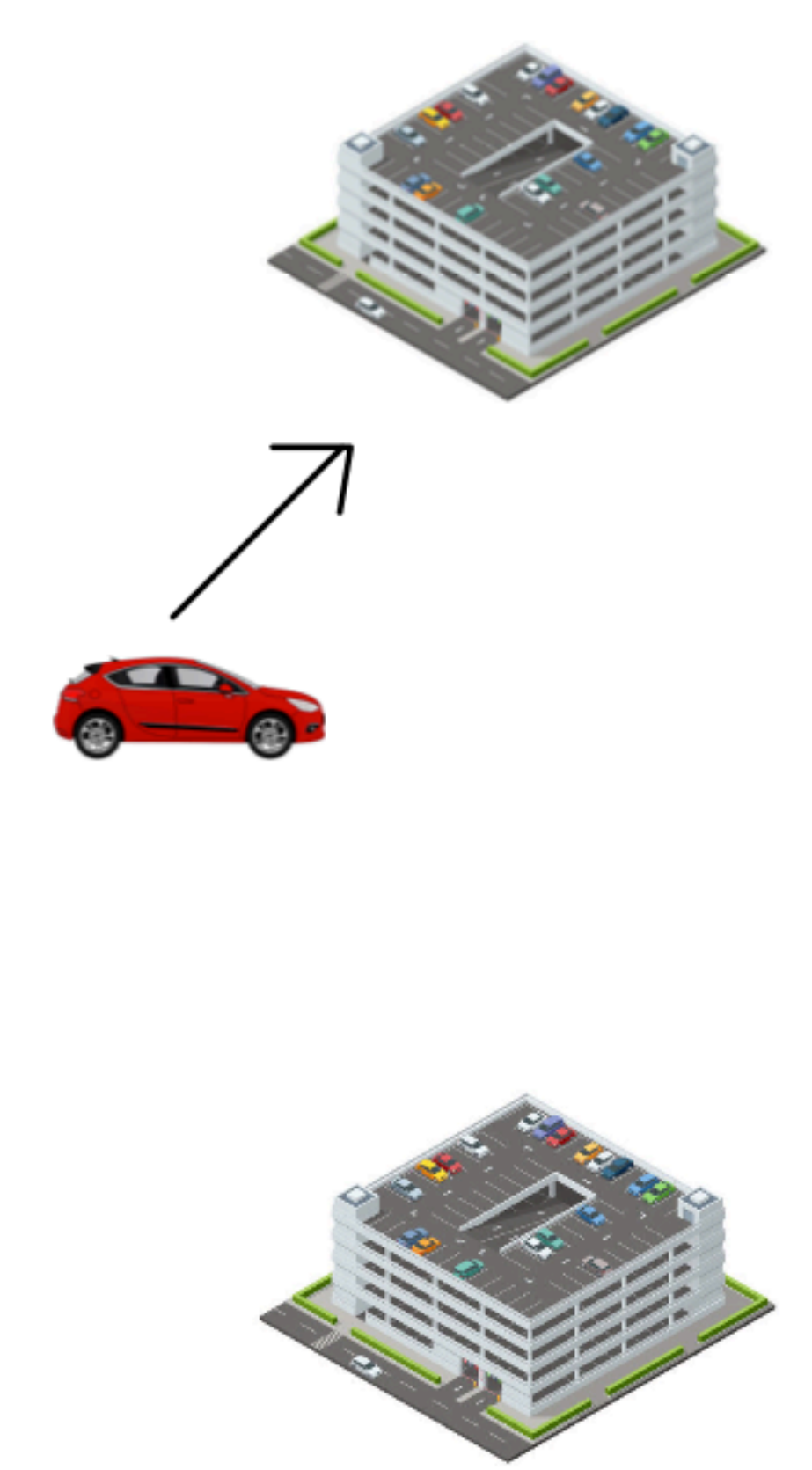












Online Transportation Problem

- **Offline Input (Setting):**
 - Metric space $M = (X, d)$
 - m parking garages with capacities $c_1, c_2, \dots, c_m$
- **Online Input:**
 - Car arrives at location $x \in X$
 - Must be assigned to garage $i \in [m]$
 - Continues until all garages full
- **Objective: Minimize** total distance cars travel

Past Work

- **Online Metric Matching**
 - $c_i = 1$ for all $i \in [m]$
 - **Thm (Kalyanasundaram, Pruhs; Khuller et. al):**
 - Greedy algorithm is (2^m-1) -competitive.
 - Retrospective / Permutation algorithm is $(2m-1)$ -competitive.
 - All deterministic algorithms have competitive ratio **at least $2m-1$** .
 - **Thm (Bansal et. al; Meyerson et. al):** $O(\log^2 m)$ -competitive randomized algorithm.

Past Work

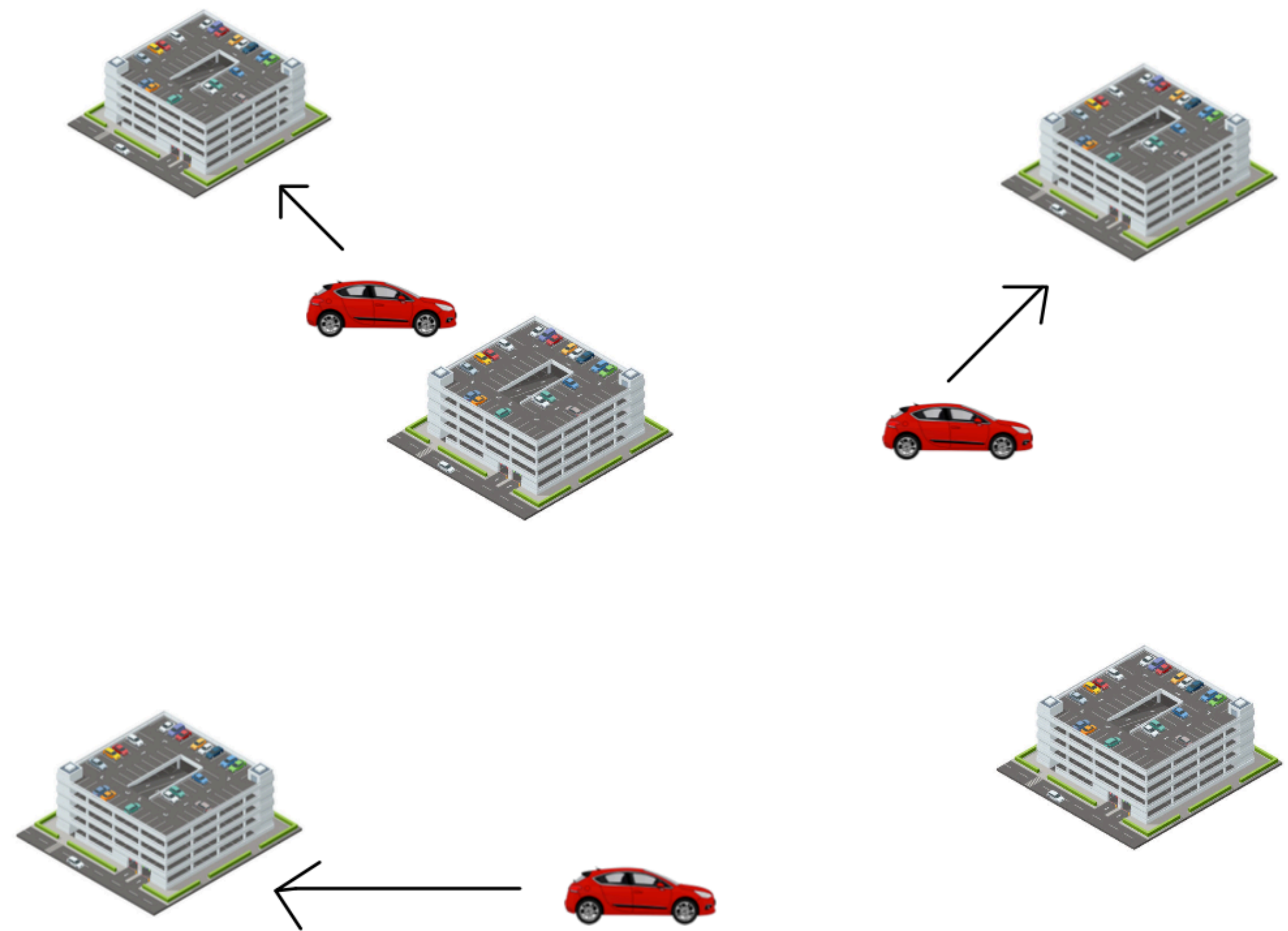
- **Online Transportation Problem**
 - General capacities
 - **Conj (Kalyanasundaram, Pruhs)**: There is a **$(2m-1)$** -competitive algorithm.
 - **Thm (Harada, Itoh; Nayyar, Raghvendra)**: Robust-Matching algorithm is $O(m \log^2 C)$ -competitive.
 - **Thm (Harada, Itoh)**: There is a **$(8m-5)$** -competitive algorithm.
 - First **$O(m)$** -competitive algorithm
 - **Thm (Kalyanasundaram, Pruhs, Stein)**: $O(\log^2 m)$ -competitive **randomized** algorithm.

Our Contribution

- **Online Transportation Problem**
 - **Conj** (**Kalyanasundaram, Pruhs**): There is a **$(2m-1)$** -competitive algorithm.
 - **Thm** (**Harada, Itoh**): There is a **$(8m-5)$** -competitive algorithm.
 - **Thm**: There is a **$(8m-7)$** -competitive “simple” algorithm + analysis.
 - **Hope**: Our simplifications help the field get down to $2m-1$.
 - **Main Tool**: Itinerant Car Problem from [Antoniadis et. al.](#)

Itinerant Car Problem

- **Offline Input (Setting):** Same as OTP
- **Online Input:** Same as OTP
- **Relaxation:** Cars can get moved later
- **Objective:** Same as OTP



Itinerant Car Problem

- **Offline Input (Setting):**
Same as OTP
- **Online Input:** Same as OTP
- **Relaxation:** Cars can get moved later
- **Objective:** Same as OTP



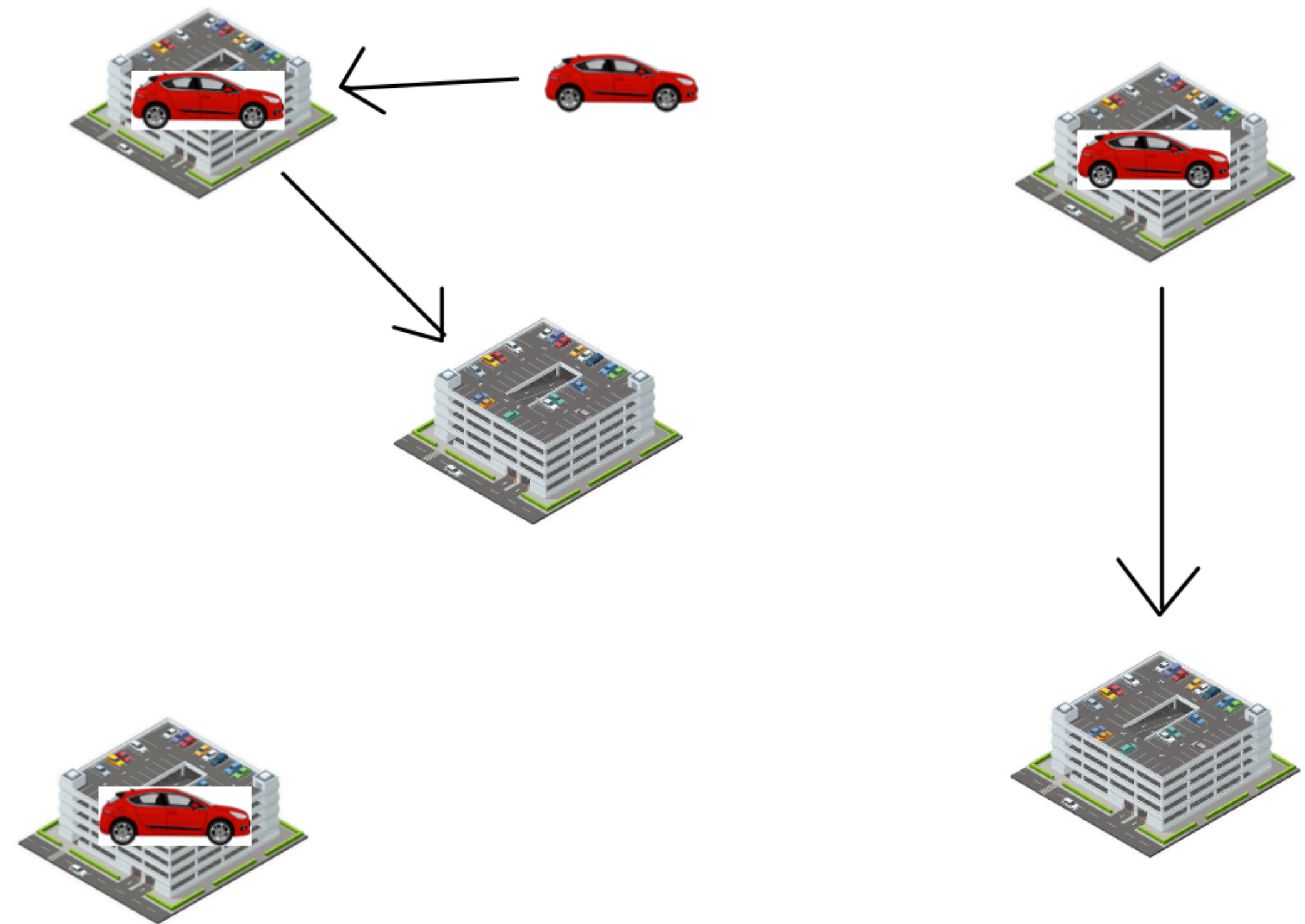
Itinerant Car Problem

- **Offline Input (Setting):**
Same as OTP
- **Online Input:** Same as OTP
- **Relaxation:** Cars can get moved later
- **Objective:** Same as OTP



Itinerant Car Problem

- **Offline Input (Setting):** Same as OTP
- **Online Input:** Same as OTP
- **Relaxation:** Cars can get moved later
- **Objective:** Same as OTP



Itinerant Car Problem

- **Offline Input (Setting):**
Same as OTP
- **Online Input:** Same as OTP
- **Relaxation:** Cars can get moved later
- **Objective:** Same as OTP





Itinerant Car

Itinerant Car

Online Transportation

Power-of-Two Tree Metrics

General Metrics

General Metrics

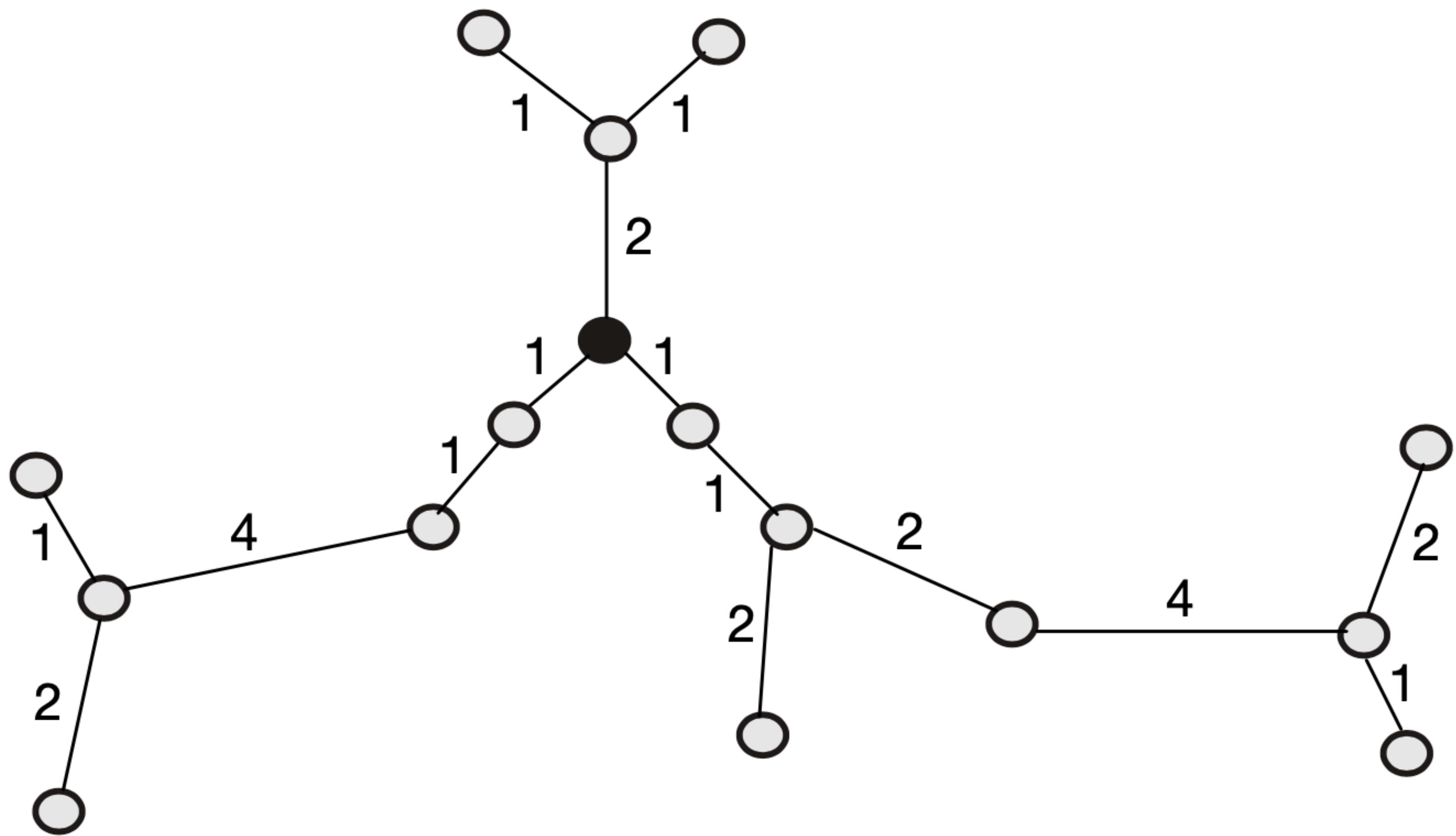
Algorithm A

Itinerant Car

Power-of-Two Tree Metrics

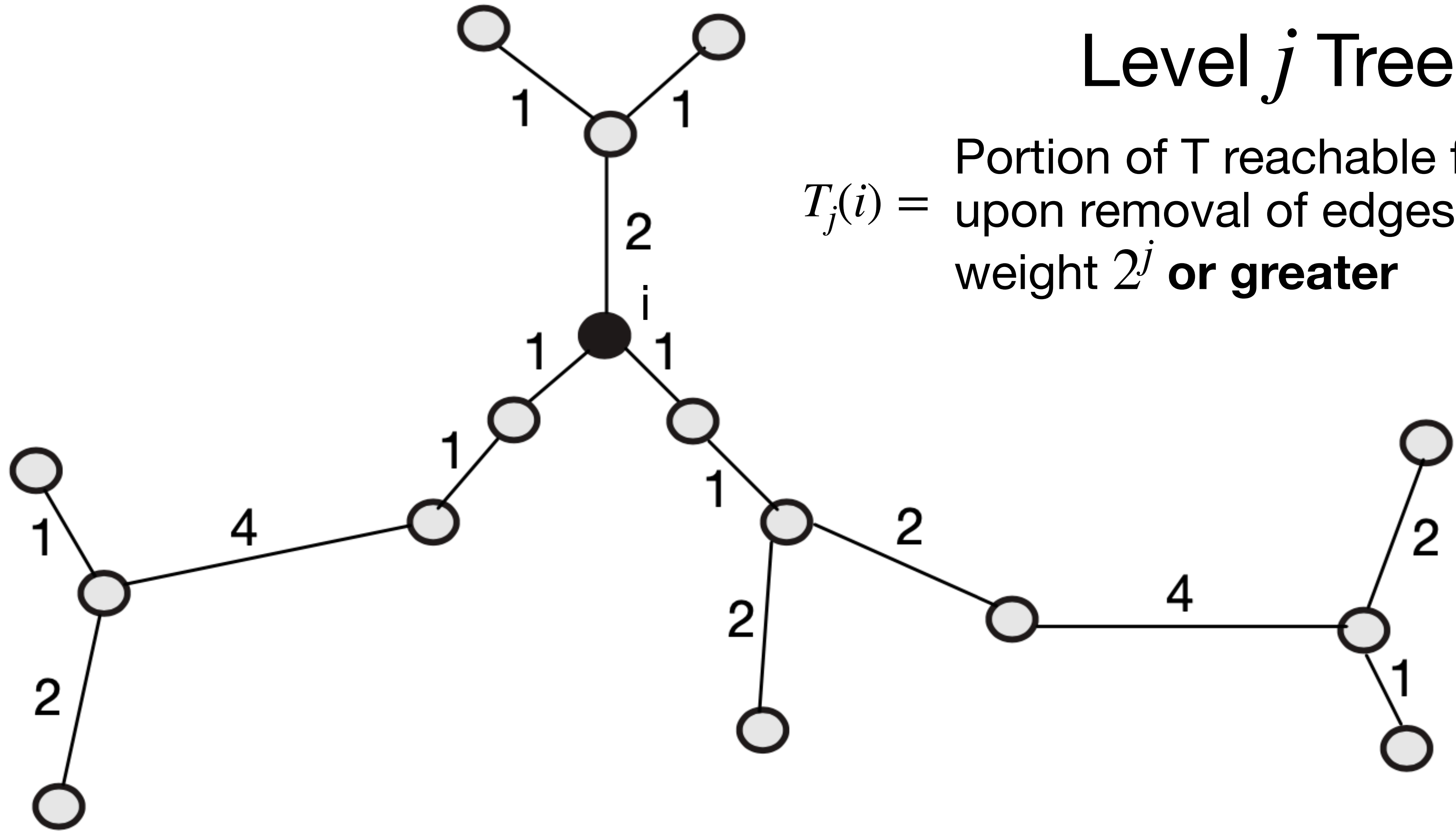
Power-of-Two Tree Metric

- Metric space defined by tree $T = (V, E)$
- Edge weights are 2^k for $k \in \mathbb{N}$
- Minimum edge weight is 1
- Garage located at **every vertex**



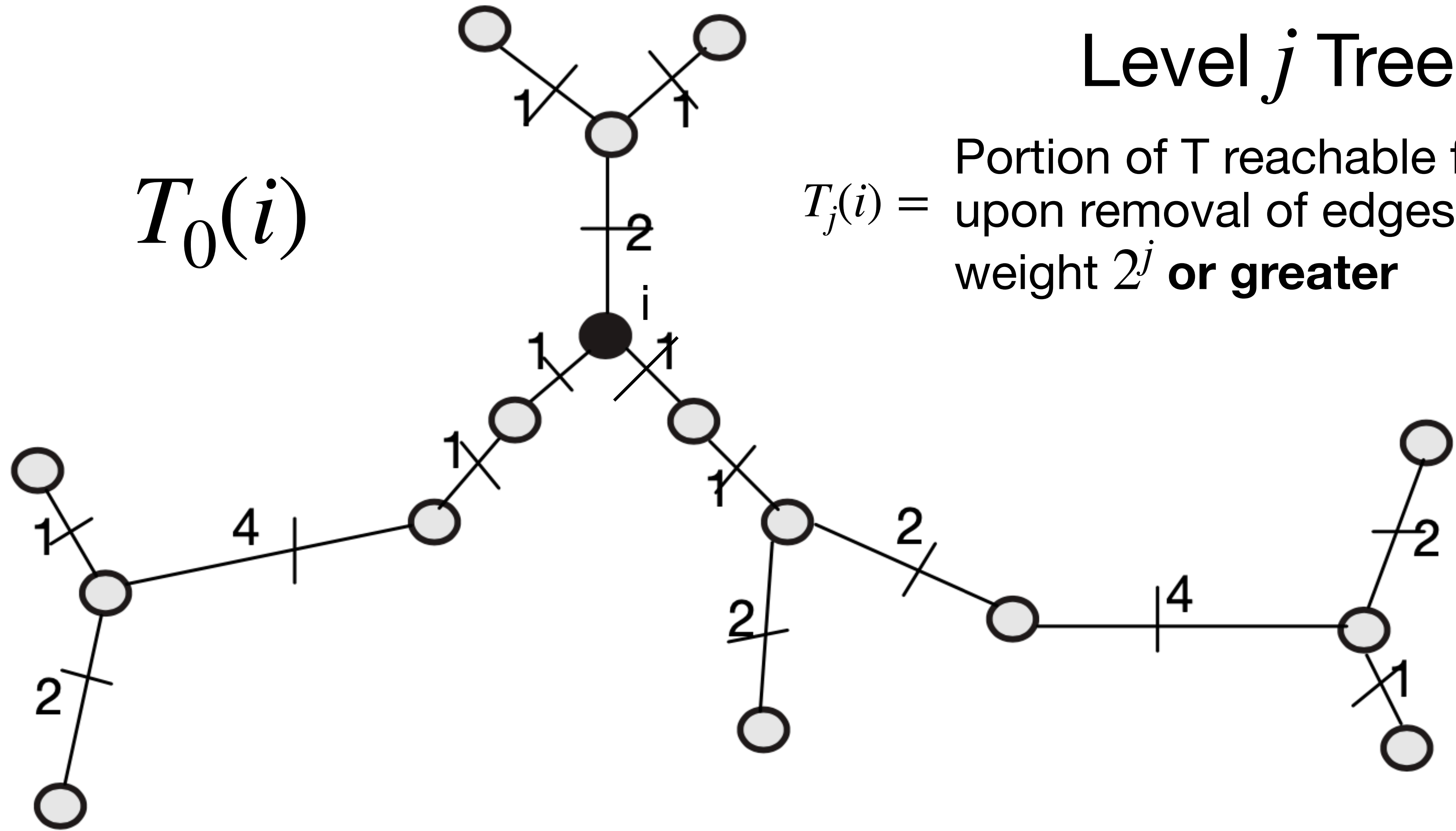
Level j Tree

$T_j(i)$ = Portion of T reachable from i
upon removal of edges with
weight 2^j or **greater**



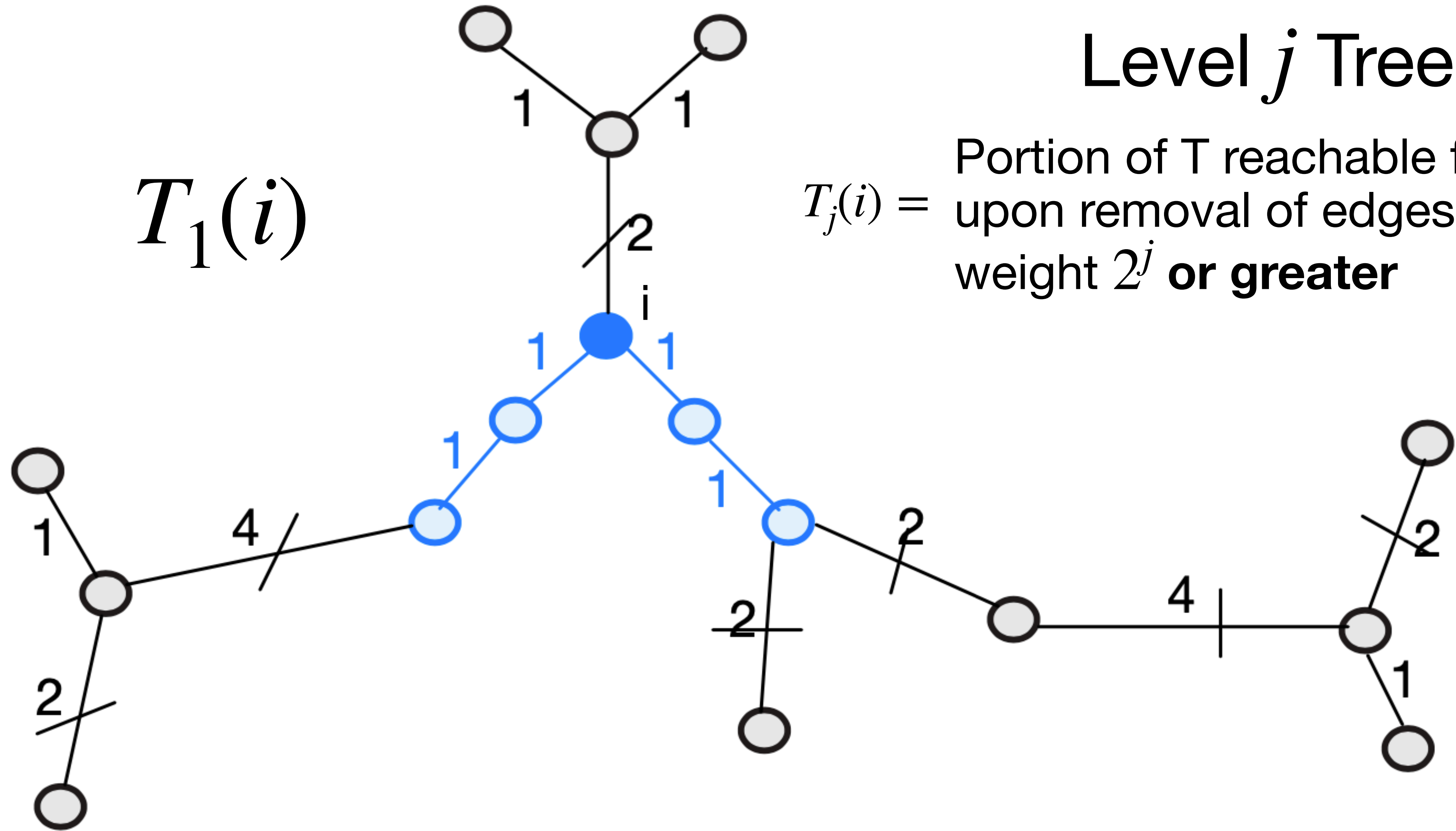
Level j Tree

$T_j(i)$ = Portion of T reachable from i
upon removal of edges with
weight 2^j or **greater**



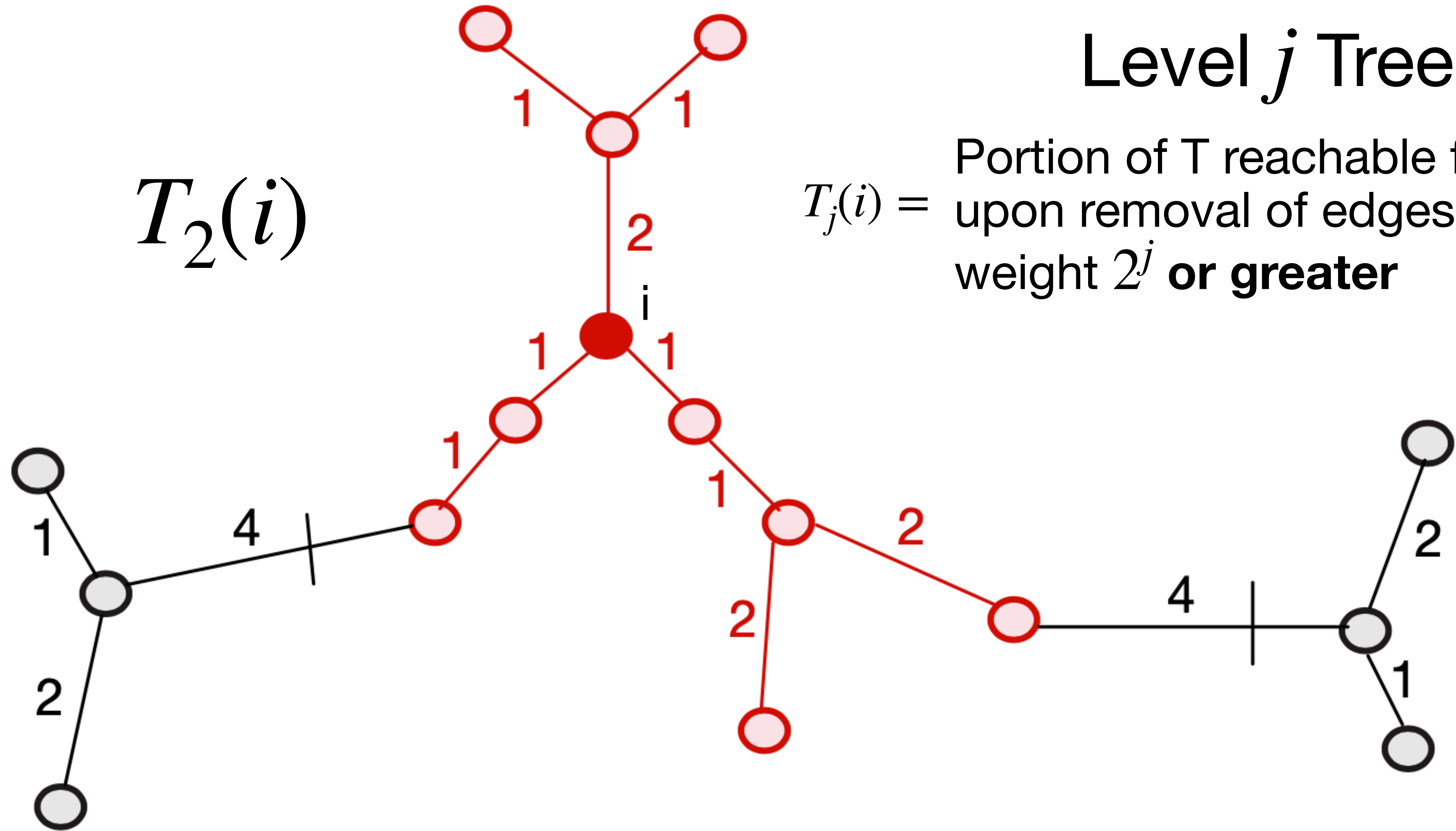
Level j Tree

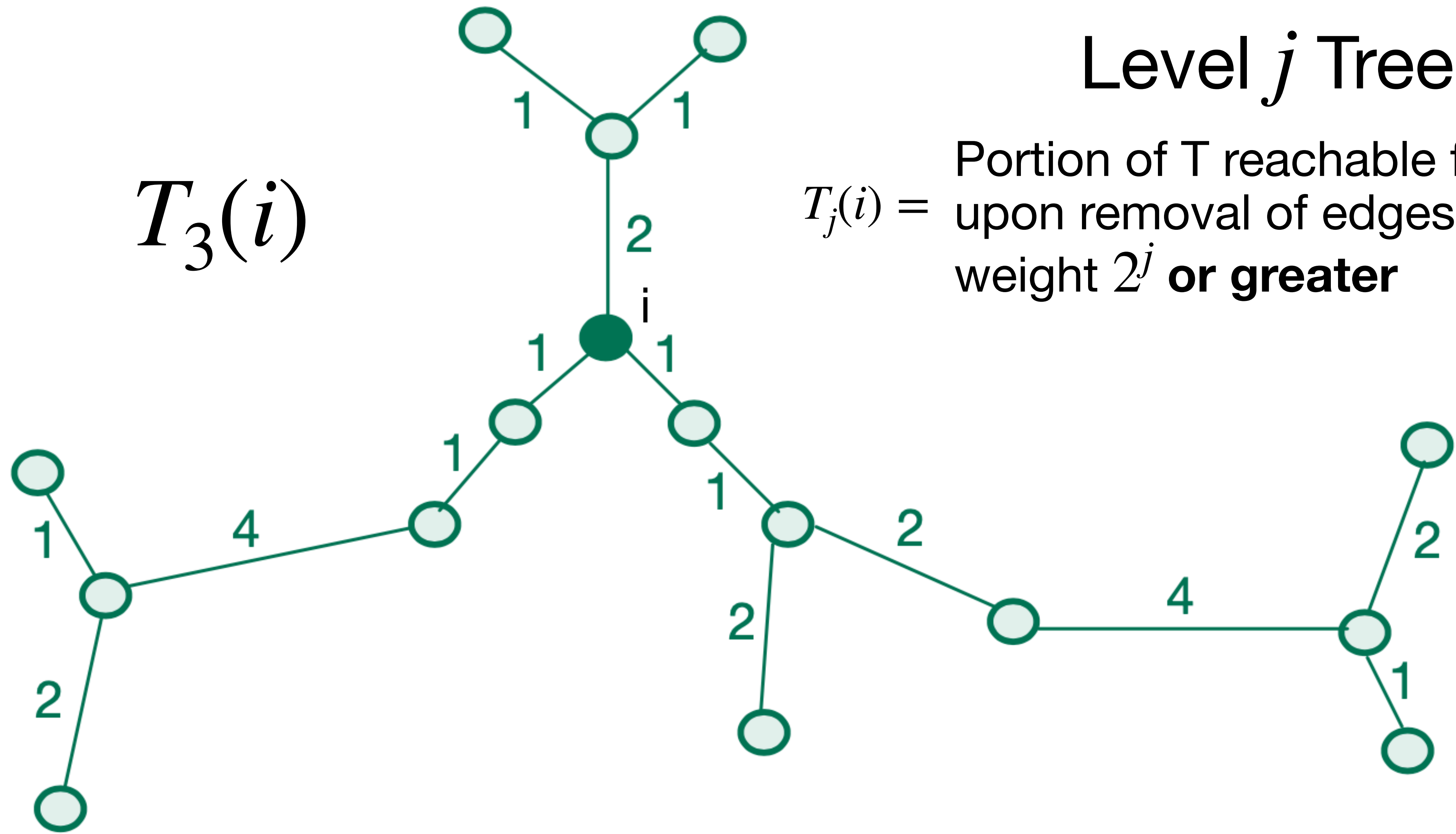
$T_j(i)$ = Portion of T reachable from i
upon removal of edges with
weight 2^j **or greater**



Level j Tree

$T_j(i)$ = Portion of T reachable from i
upon removal of edges with
weight 2^j or **greater**





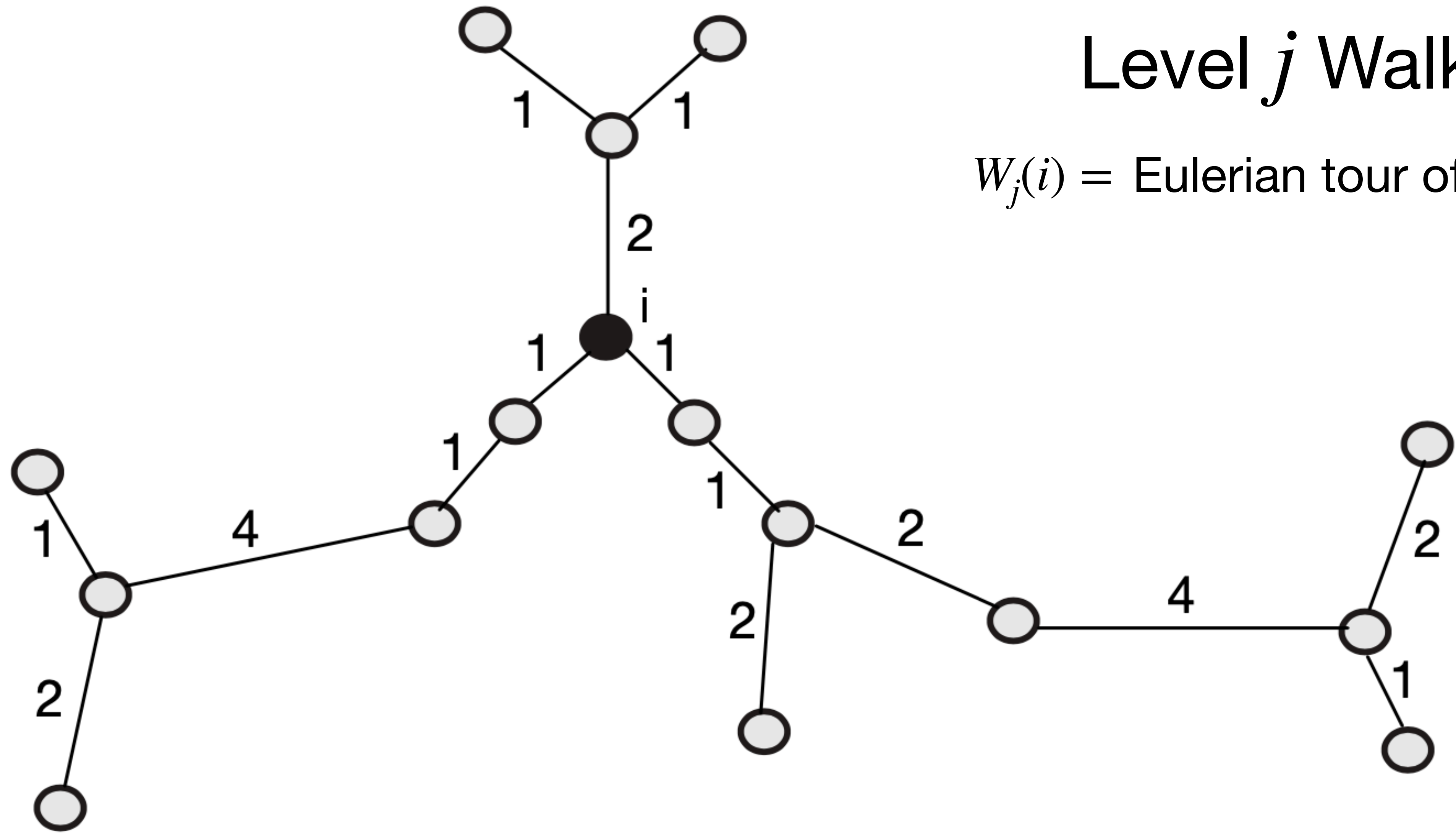
$T_3(i)$

Level j Tree

$T_j(i)$ = Portion of T reachable from i
upon removal of edges with
weight 2^j or **greater**

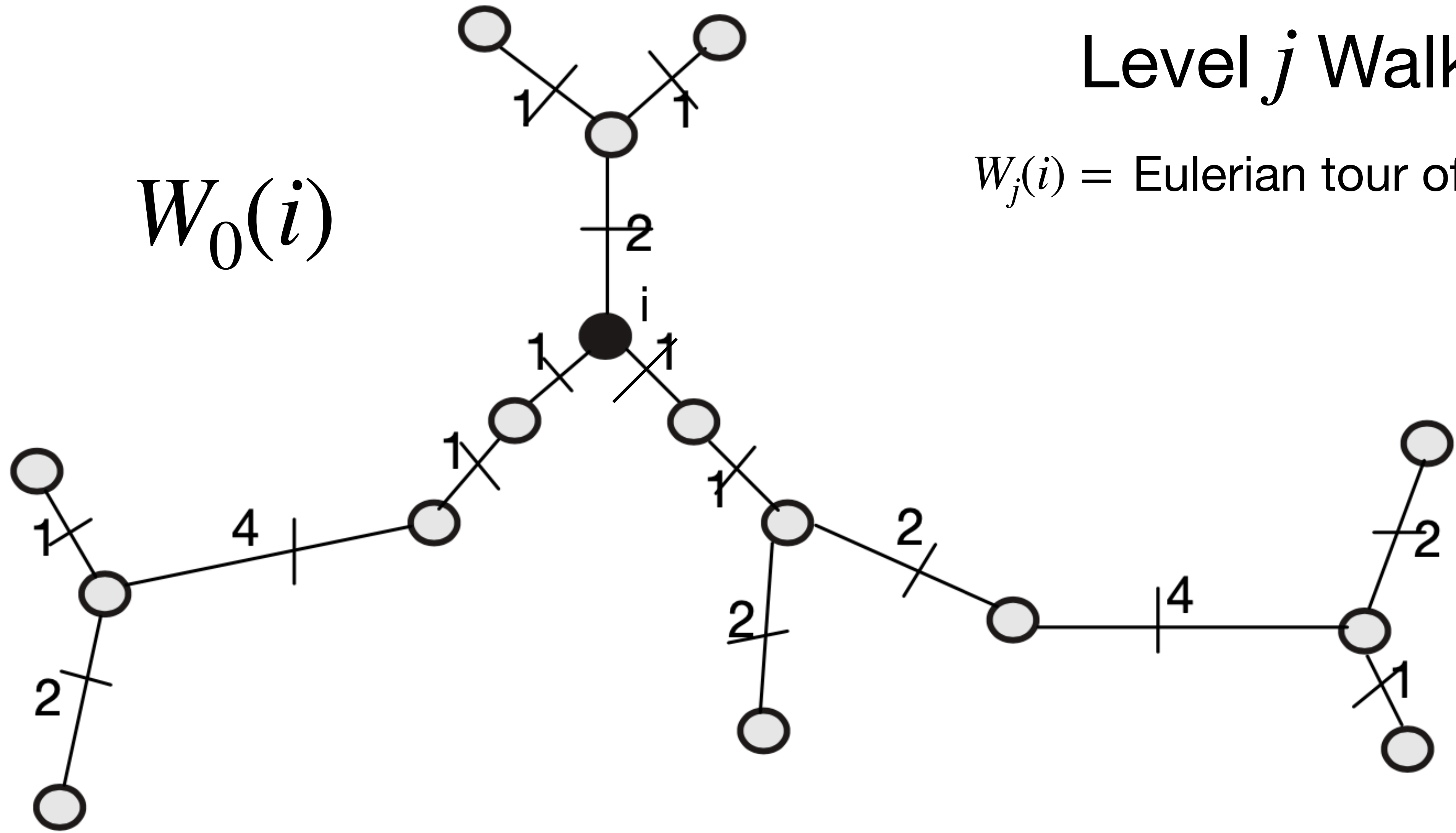
Level j Walk

$W_j(i)$ = Eulerian tour of $T_j(i)$



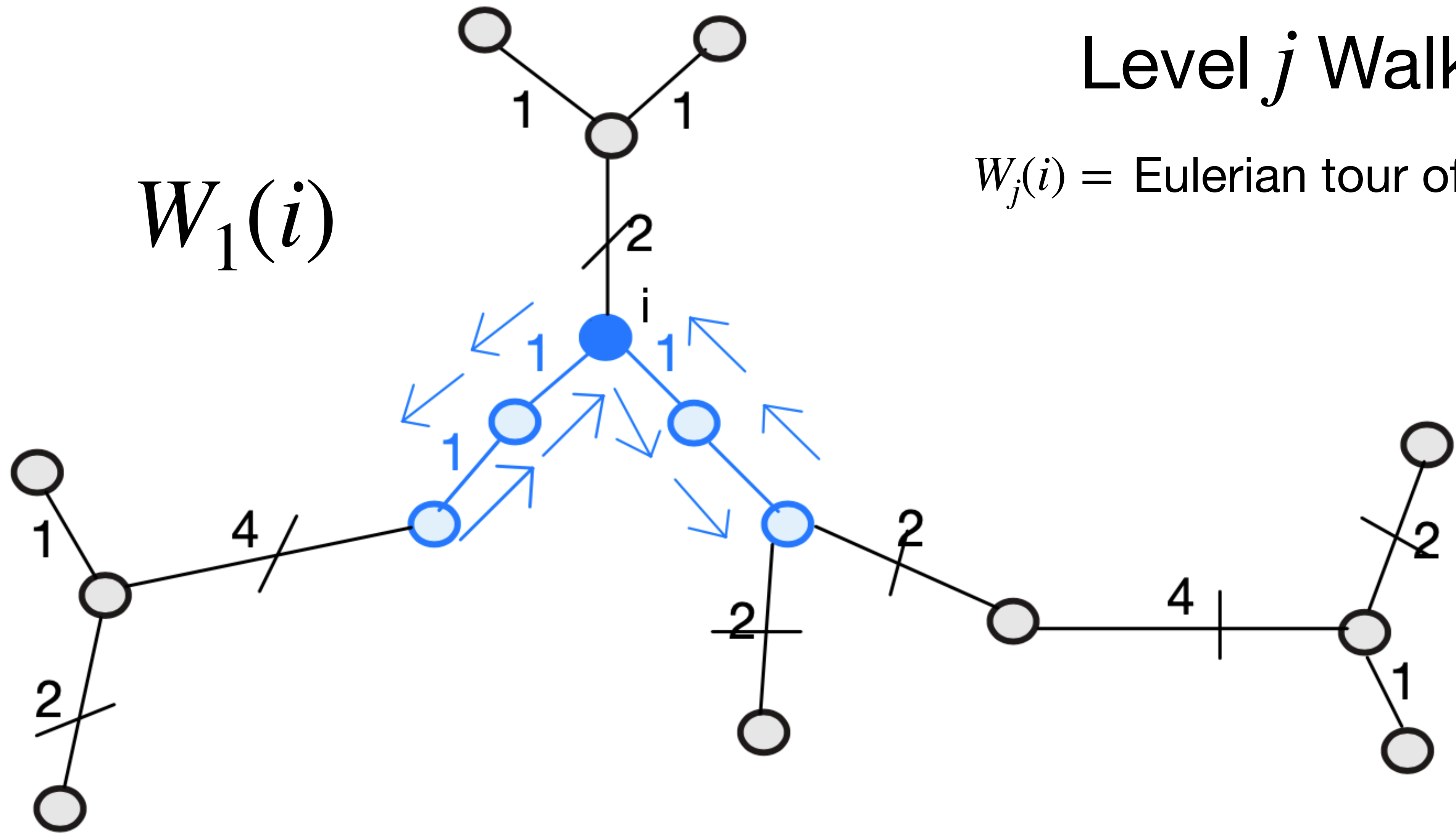
Level j Walk

$W_j(i)$ = Eulerian tour of $T_j(i)$



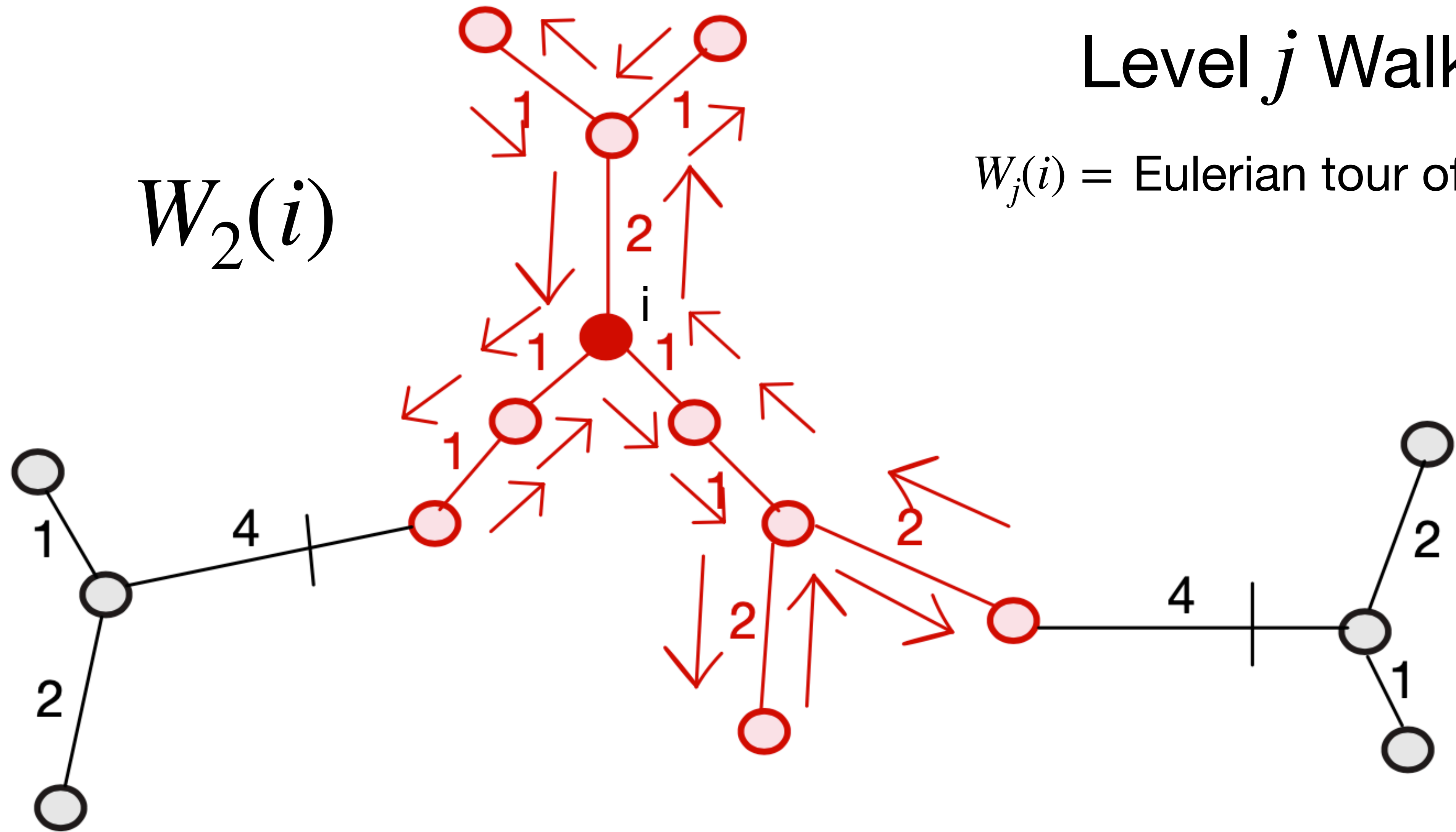
Level j Walk

$W_j(i) = \text{Eulerian tour of } T_j(i)$



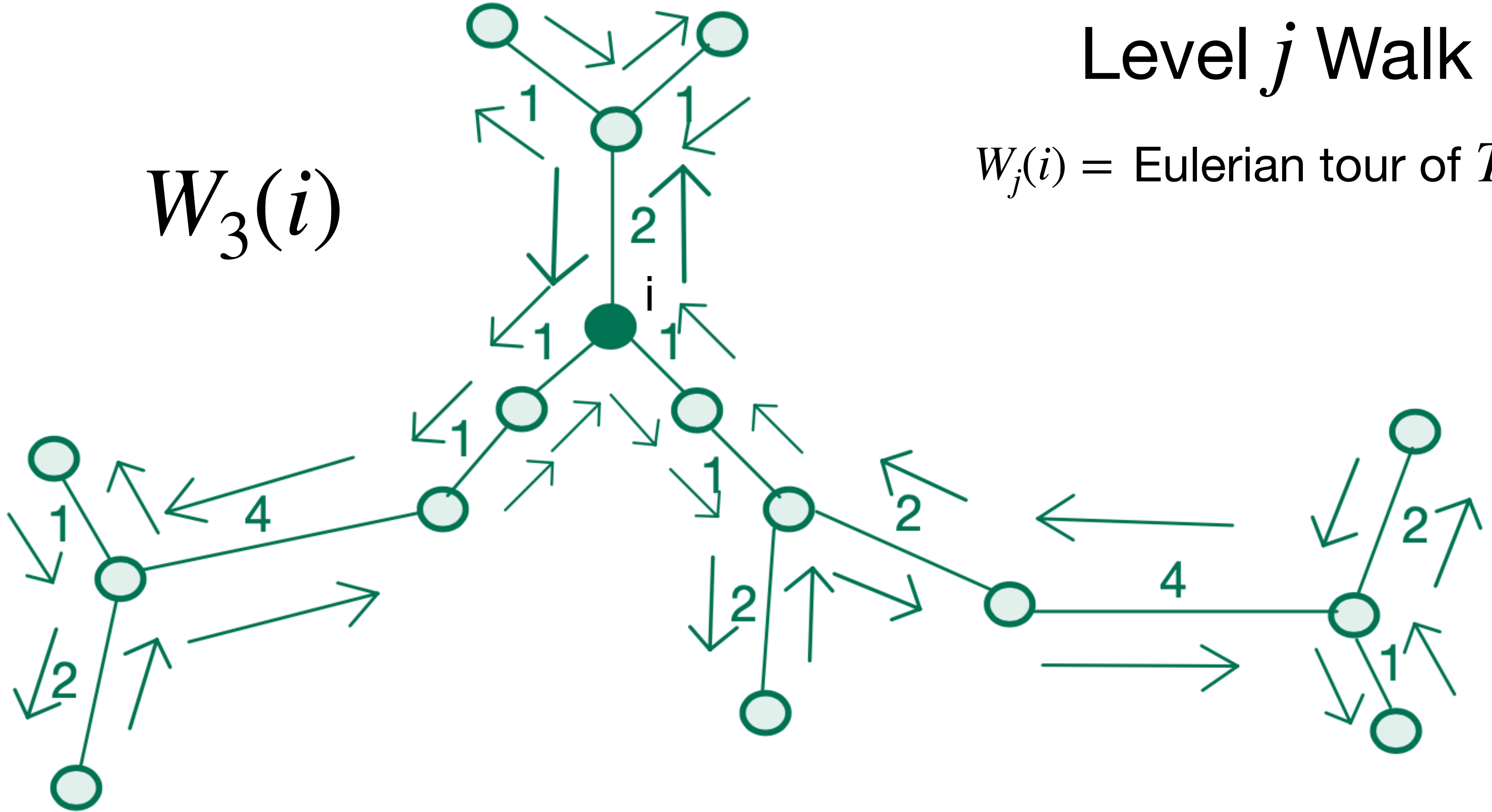
Level j Walk

$W_j(i) = \text{Eulerian tour of } T_j(i)$



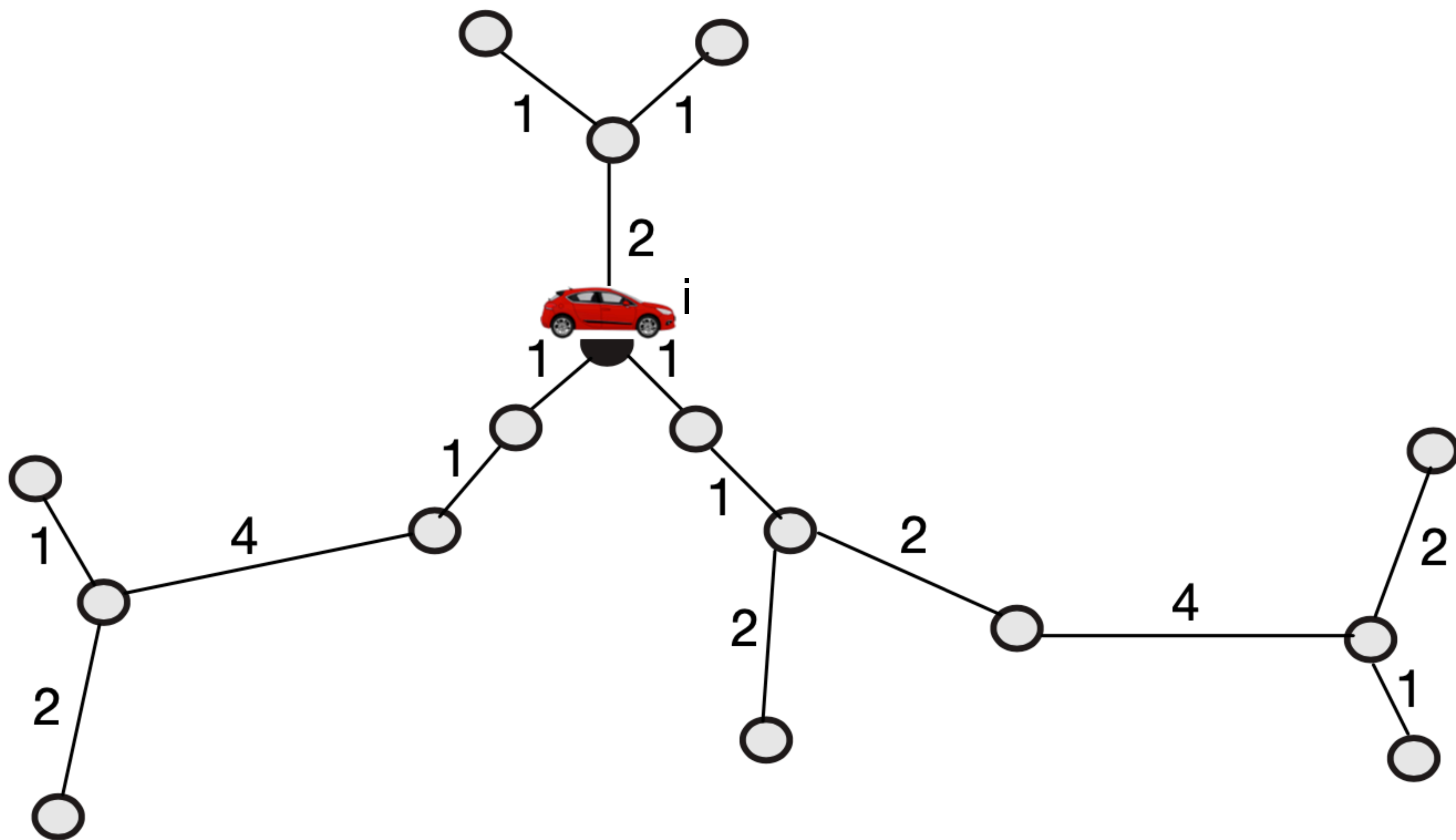
Level j Walk

$W_j(i)$ = Eulerian tour of $T_j(i)$

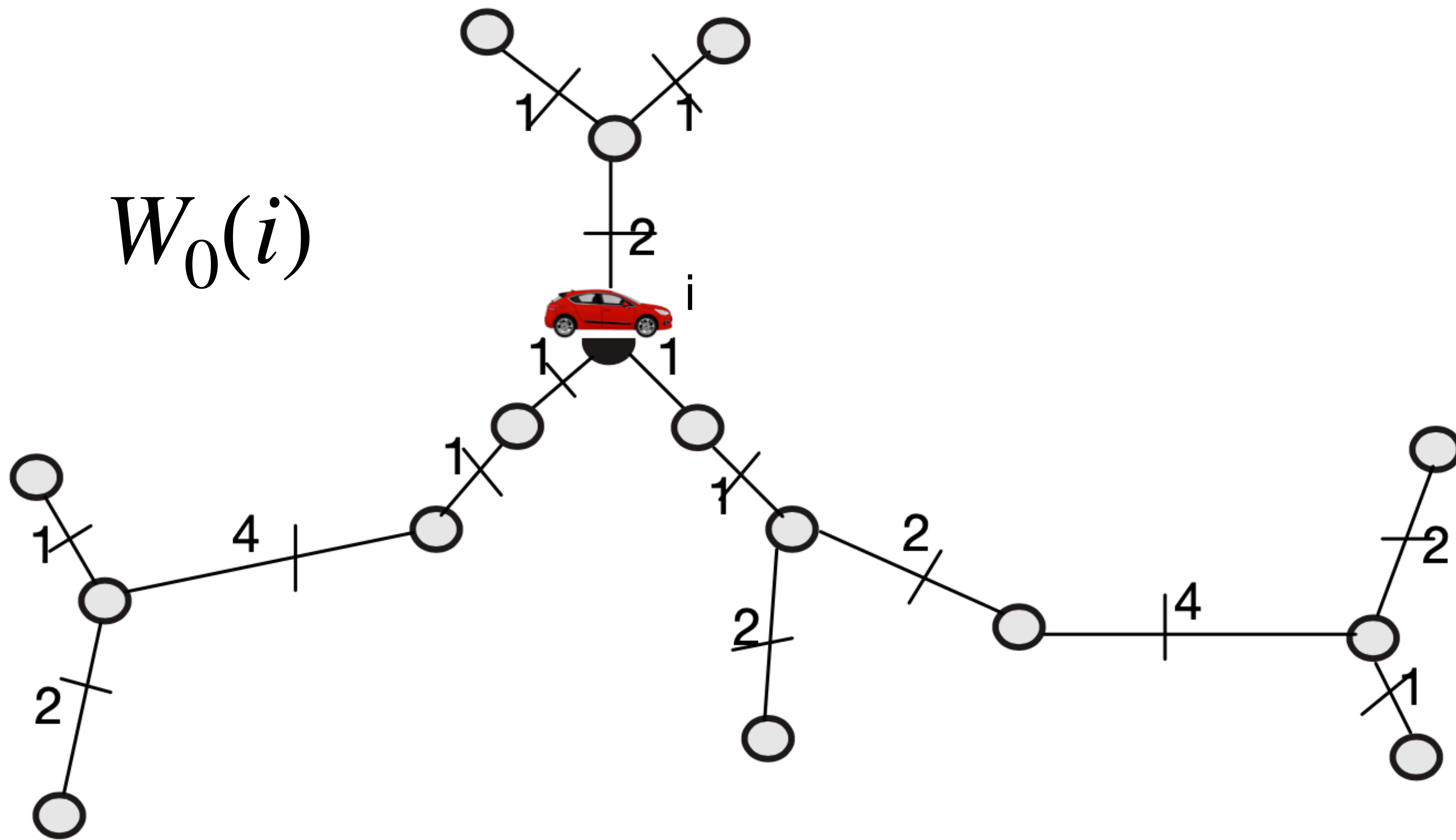


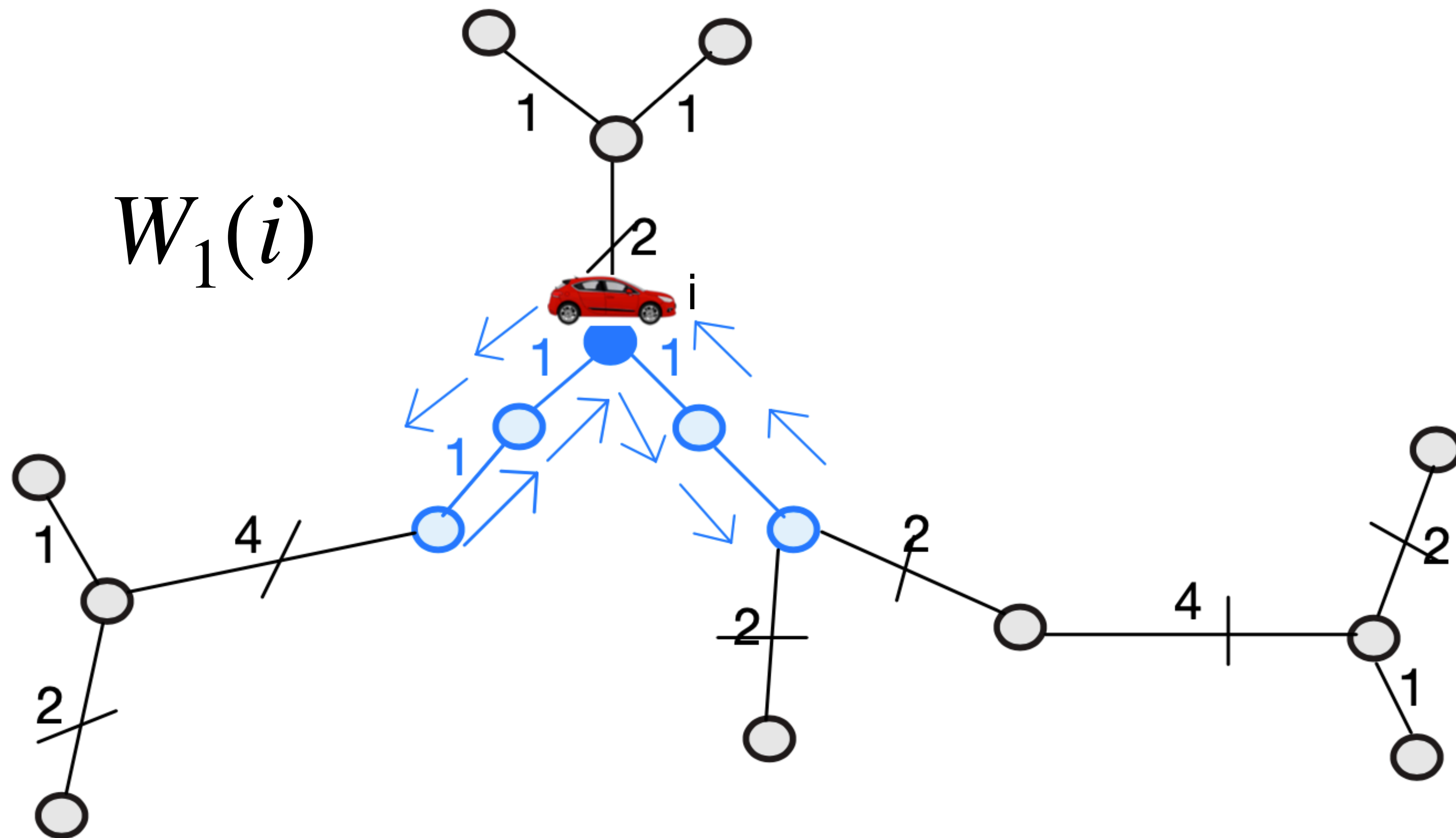
Algorithm A

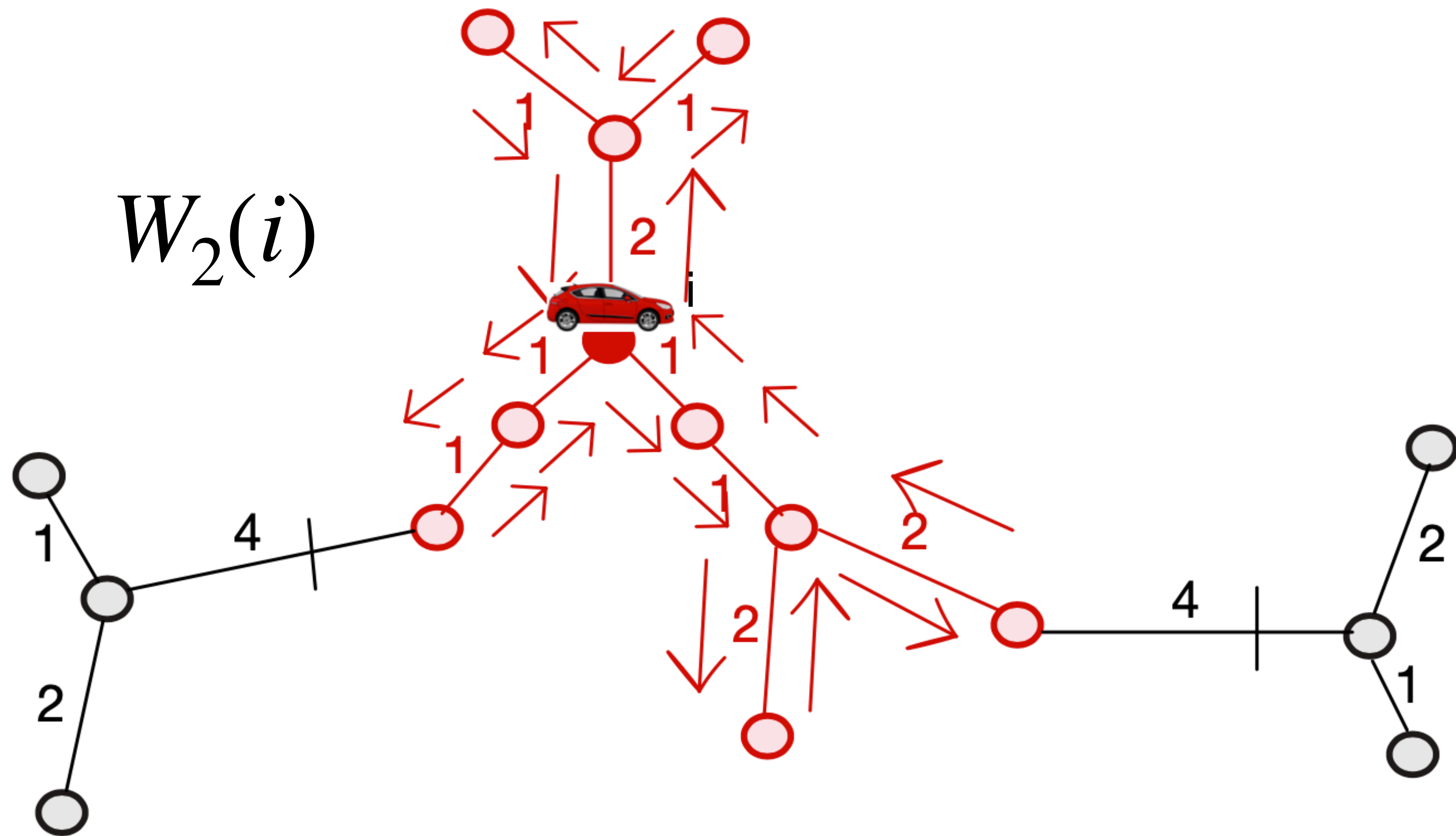
- **Recall:** Itinerant Car Problem on Power-of-Two Tree Metrics
- Each car drives along $W_0(i), W_1(i), W_2(i), \dots$ where $i =$ arrival vertex
- Upon finding a garage...
 - **If unfull:** Park there
 - **Else full:** Park there if and only if **higher-level car** in garage



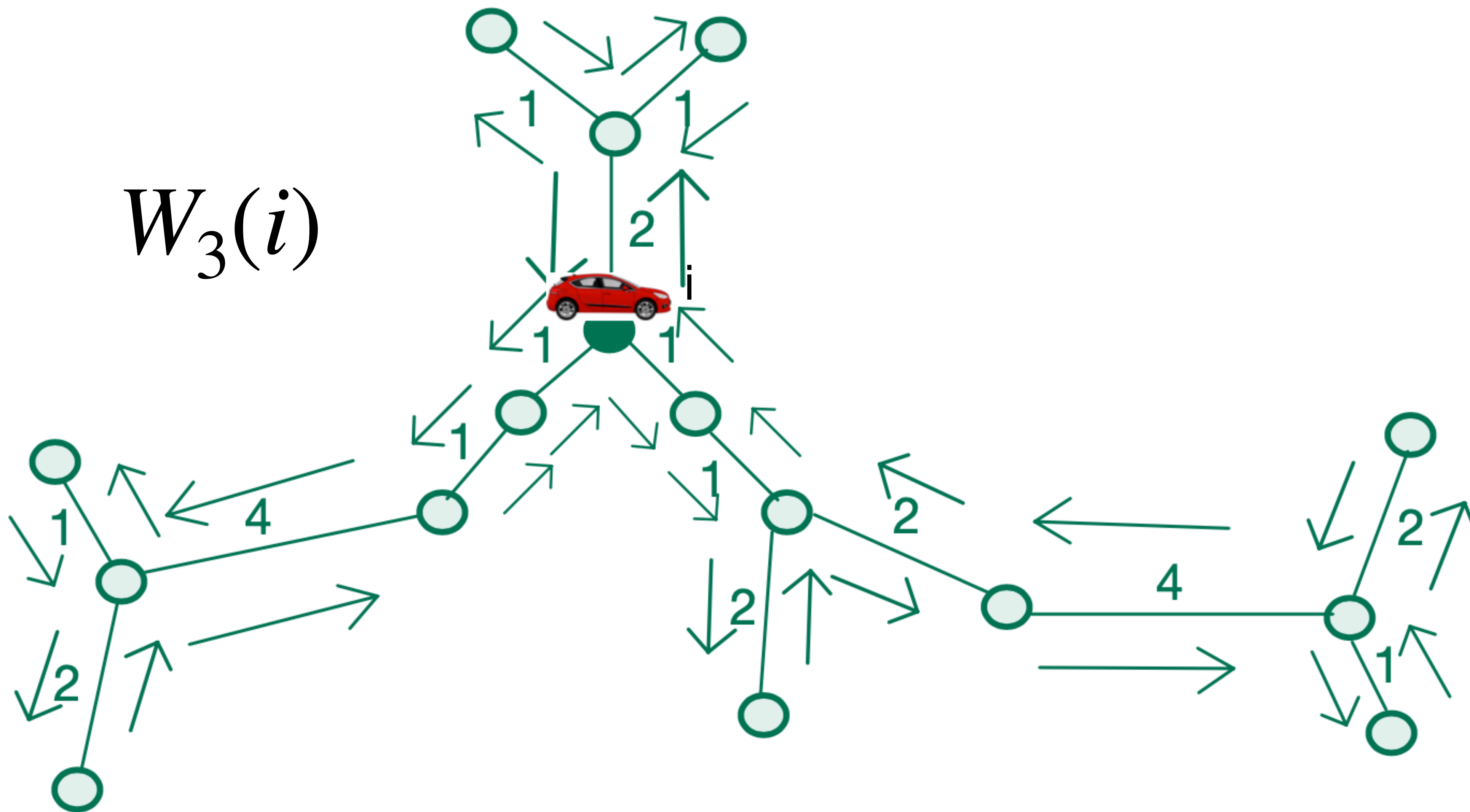
$$W_0(i)$$

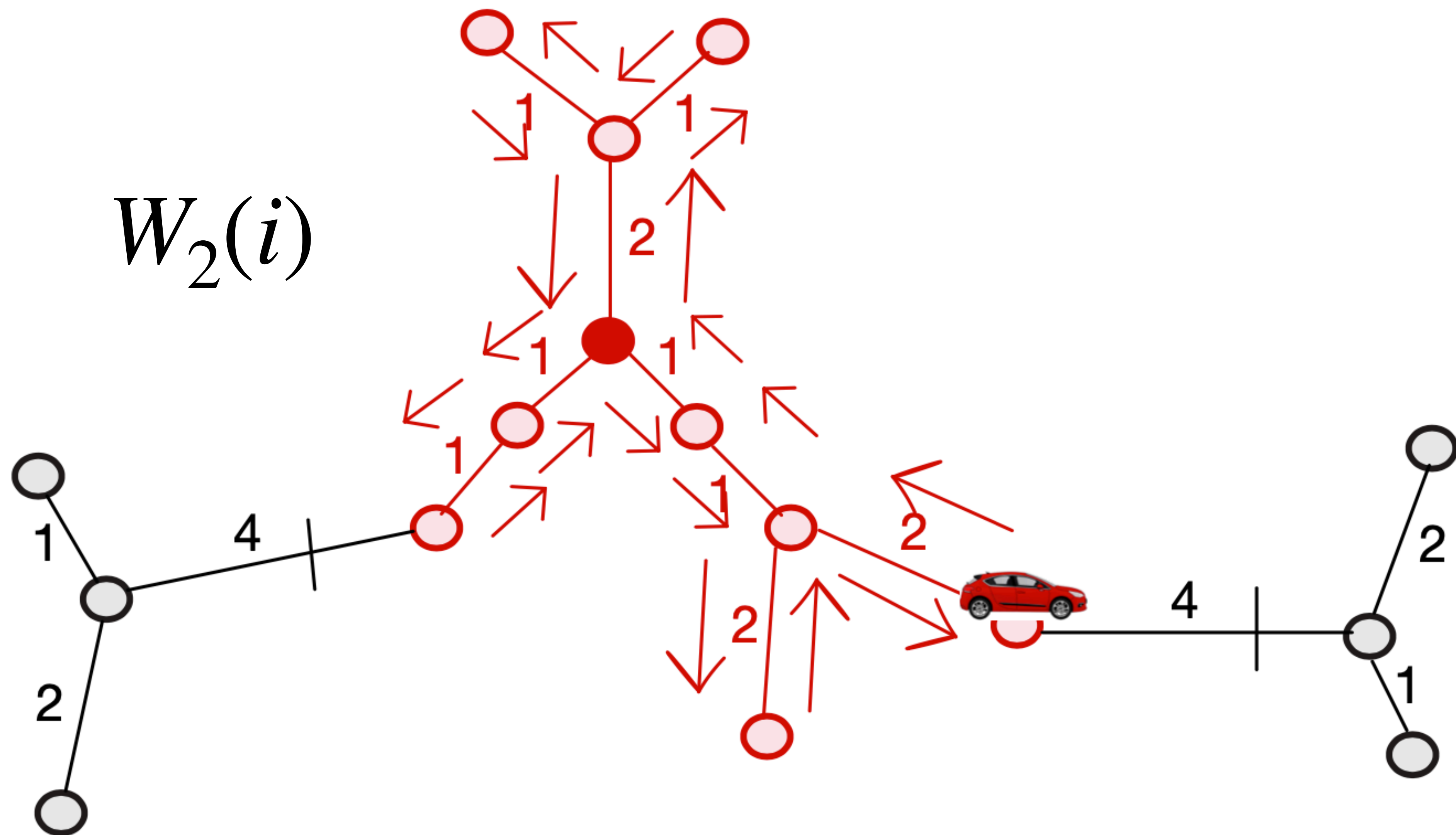


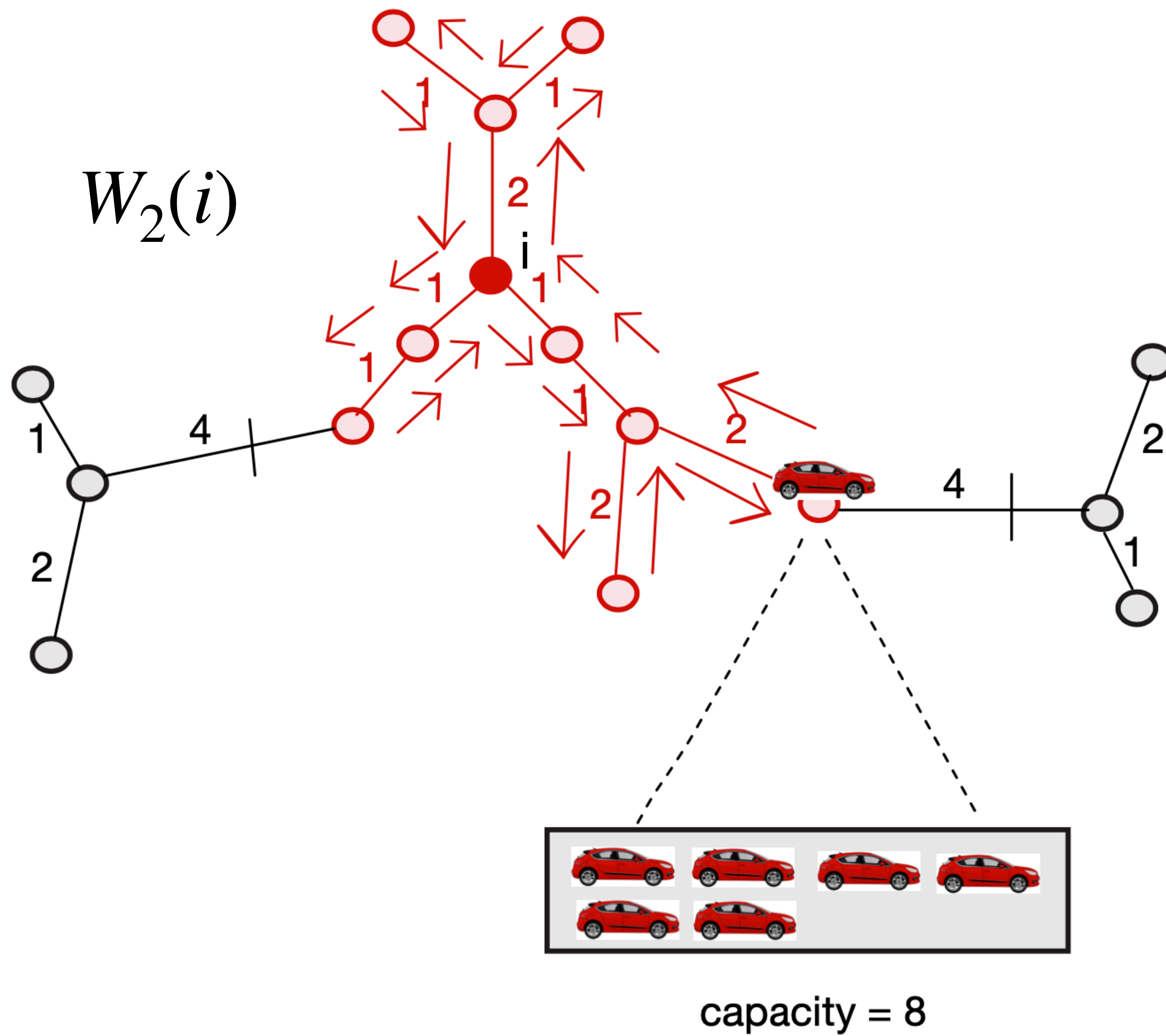


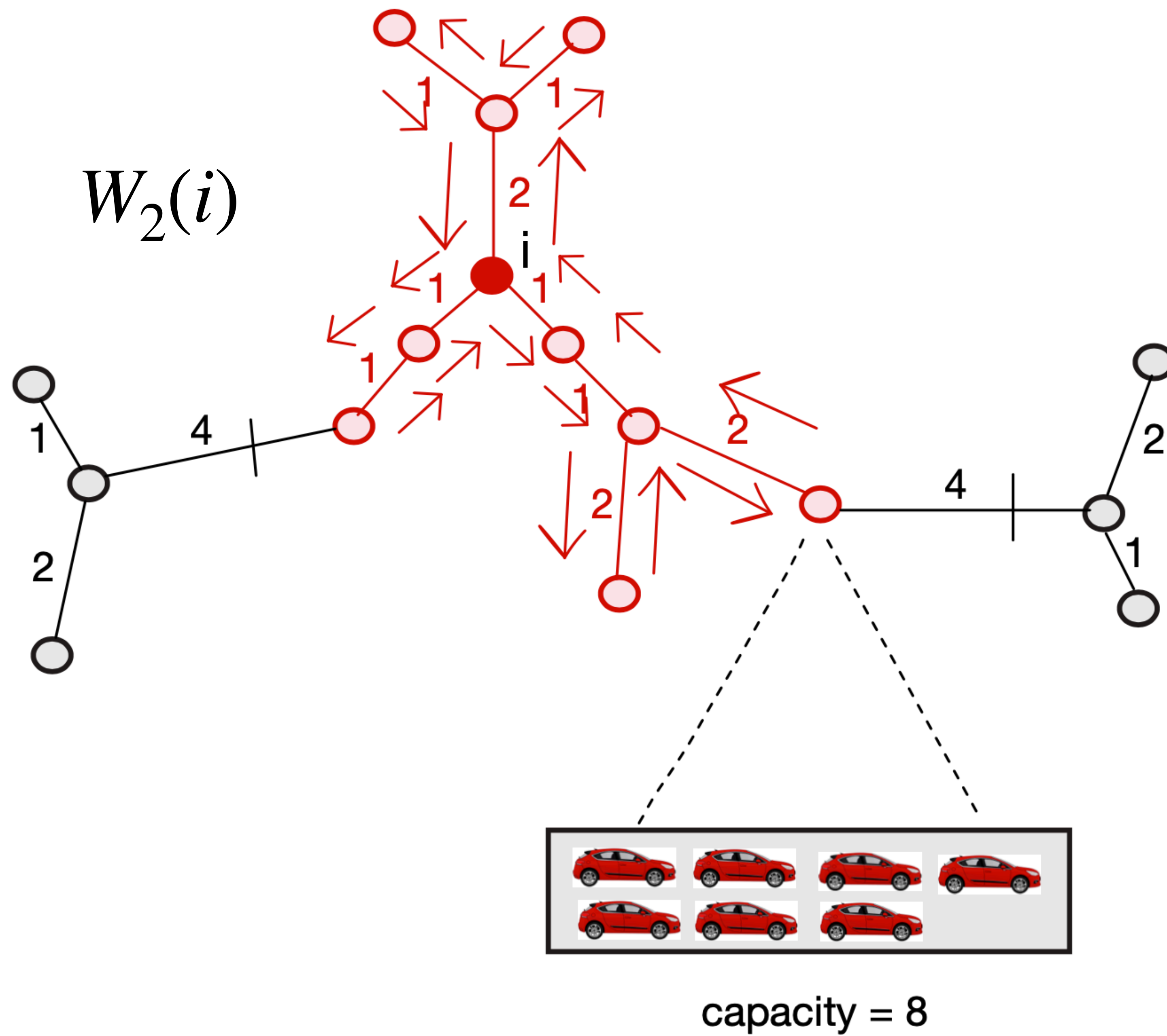


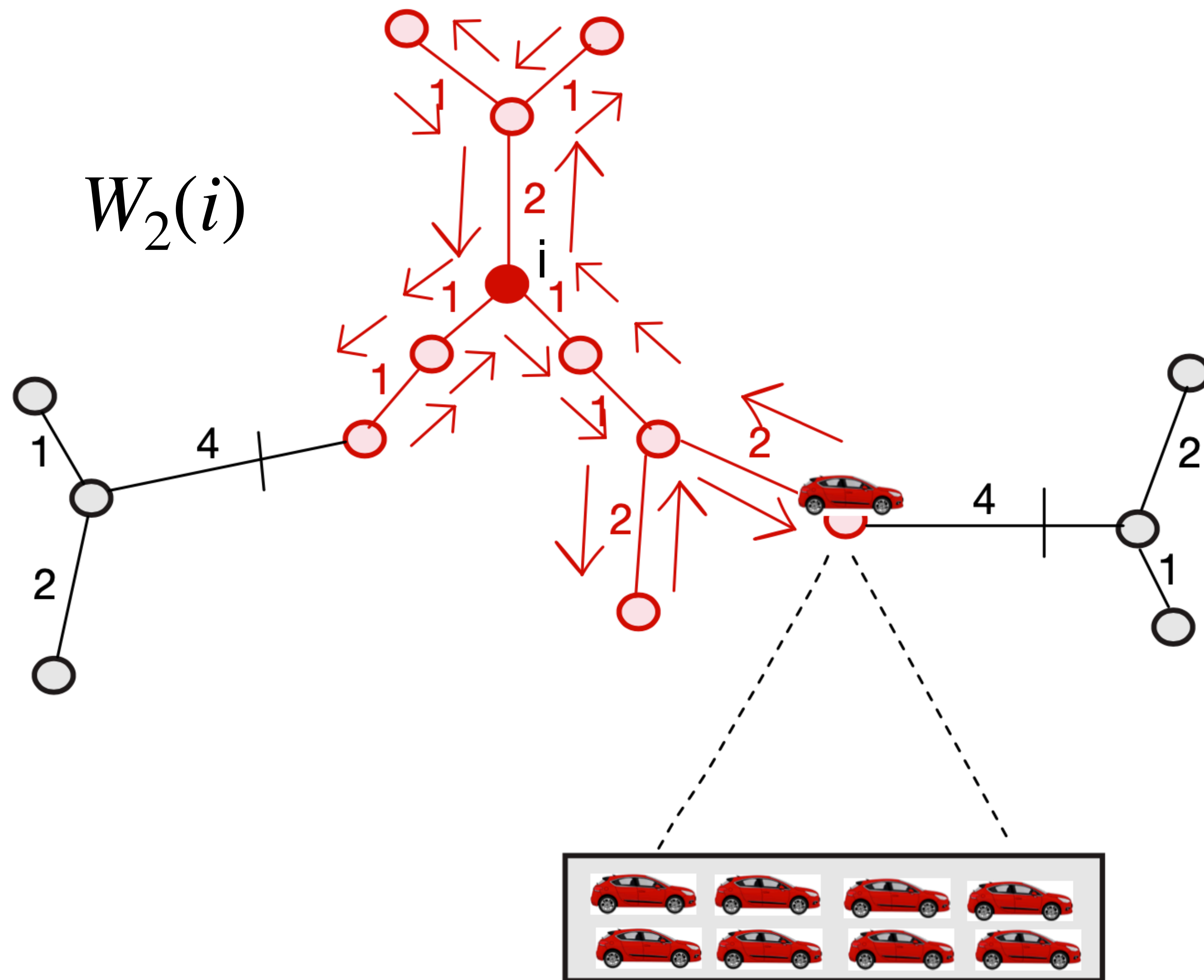
$W_3(i)$



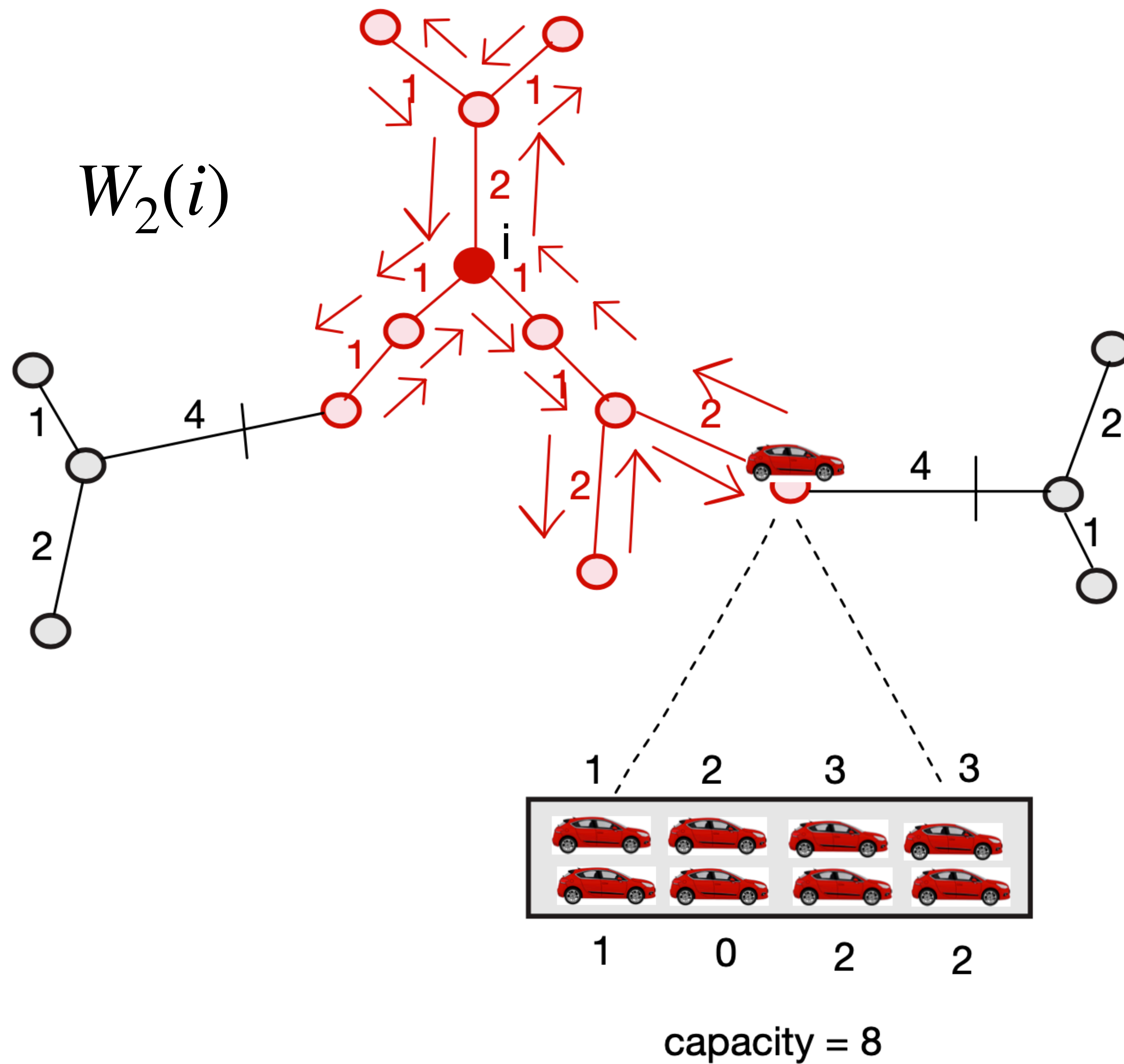


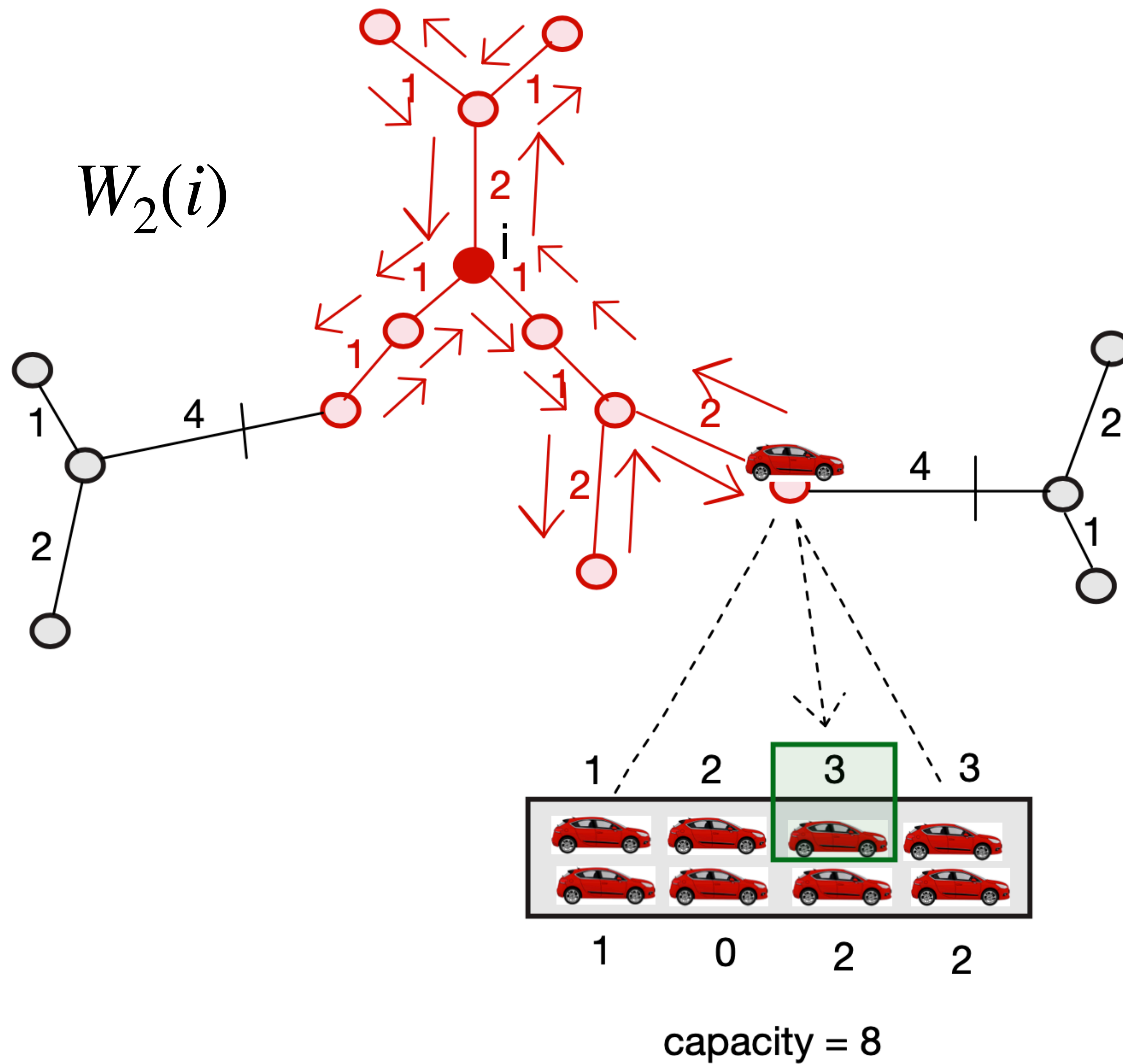


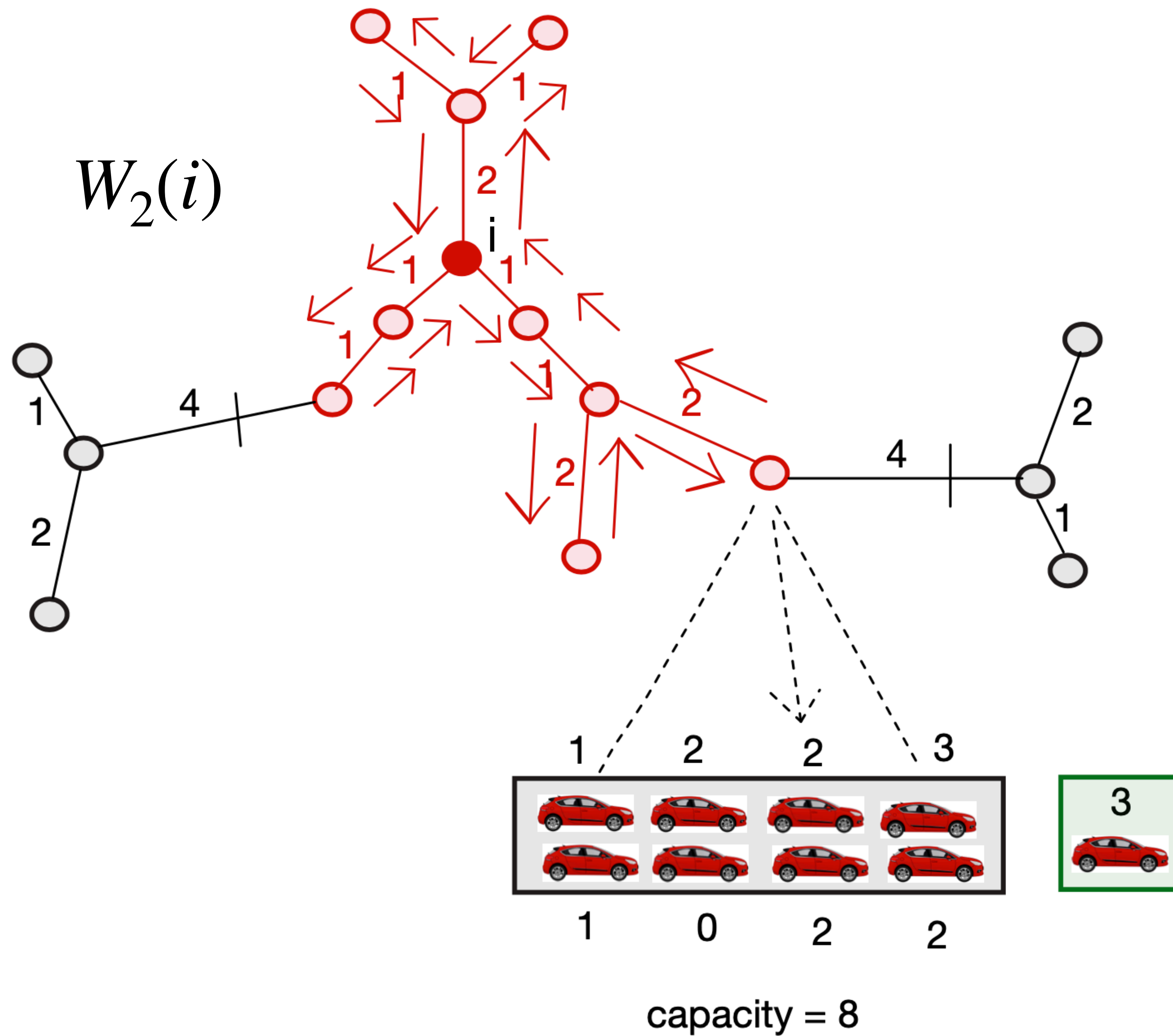


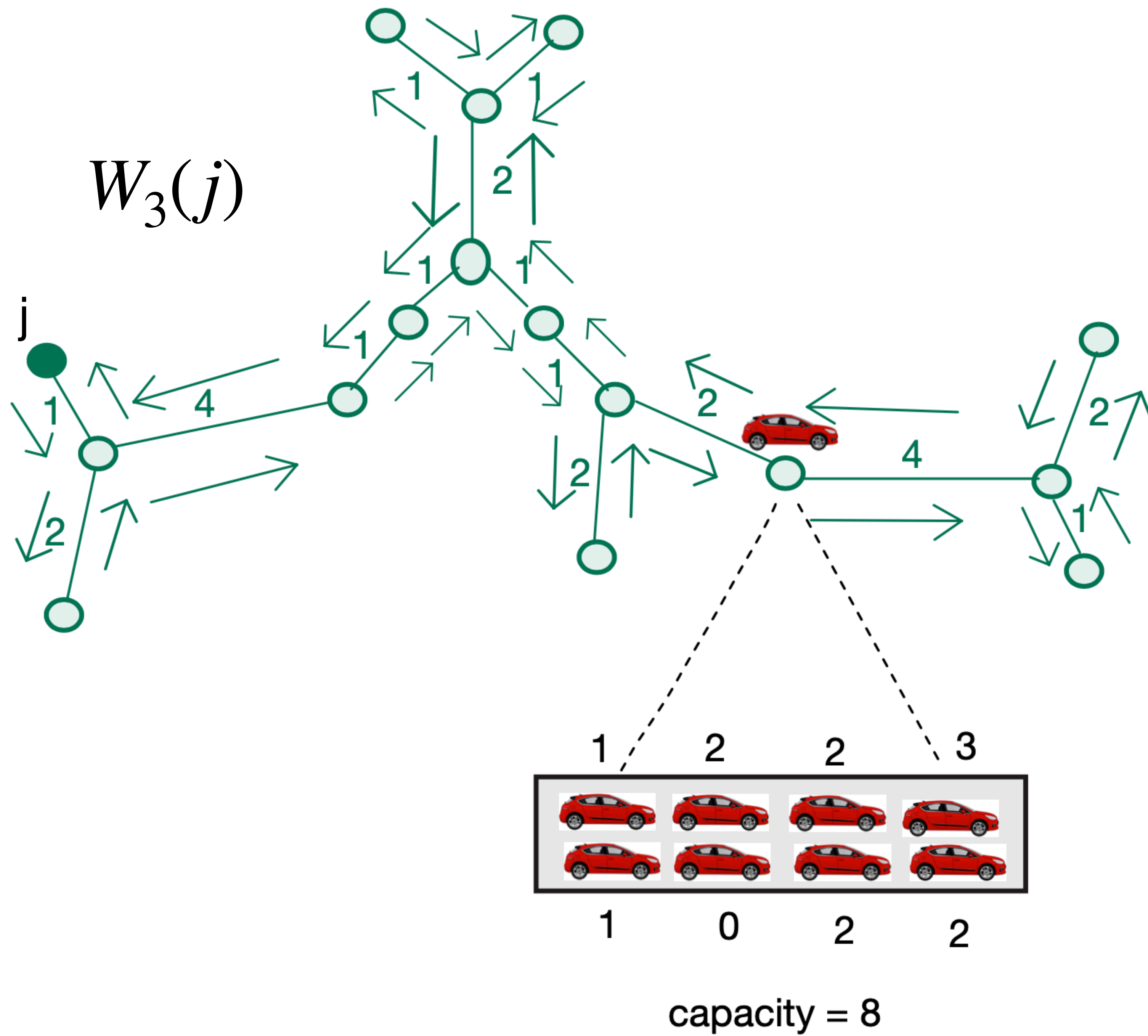


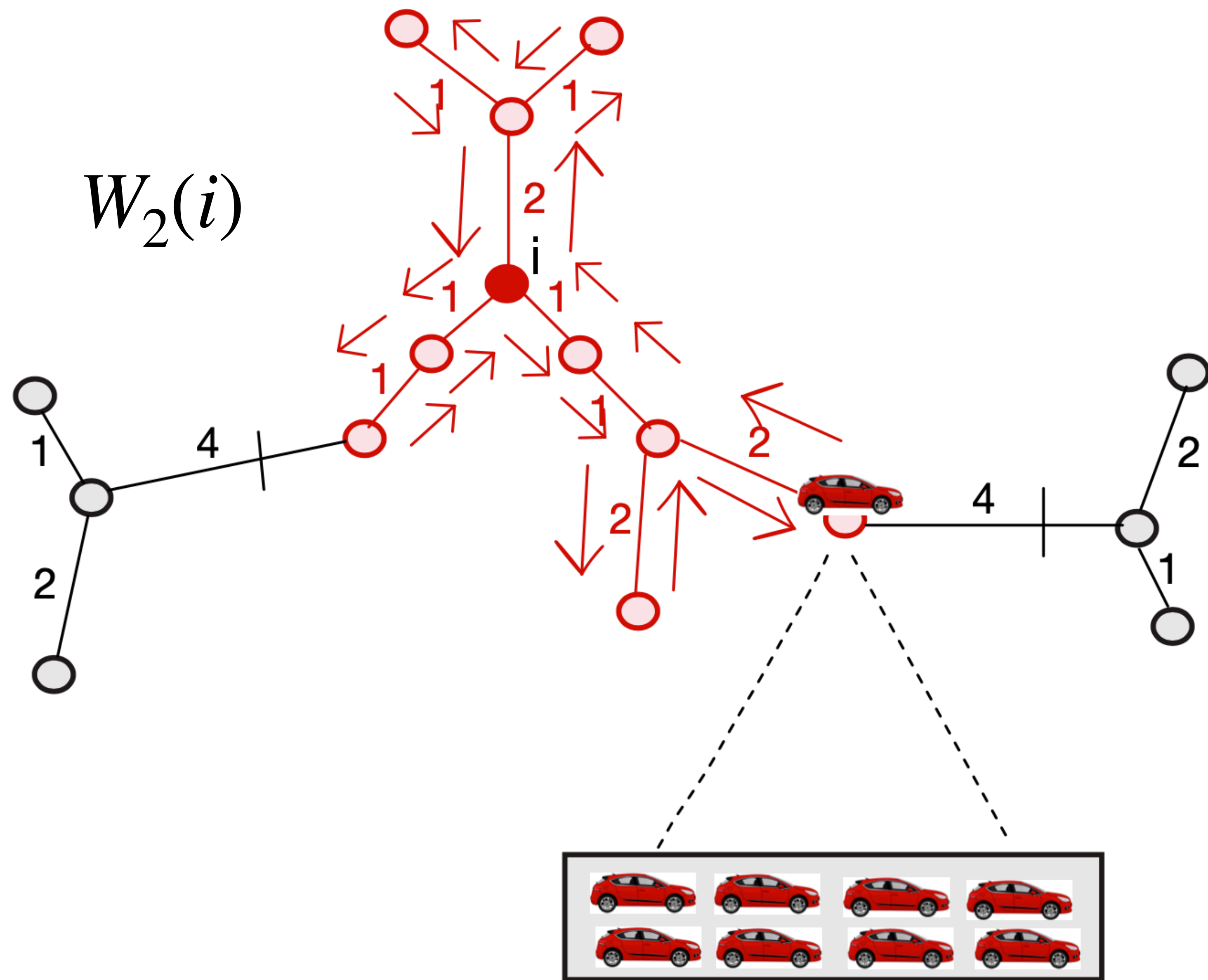
capacity = 8



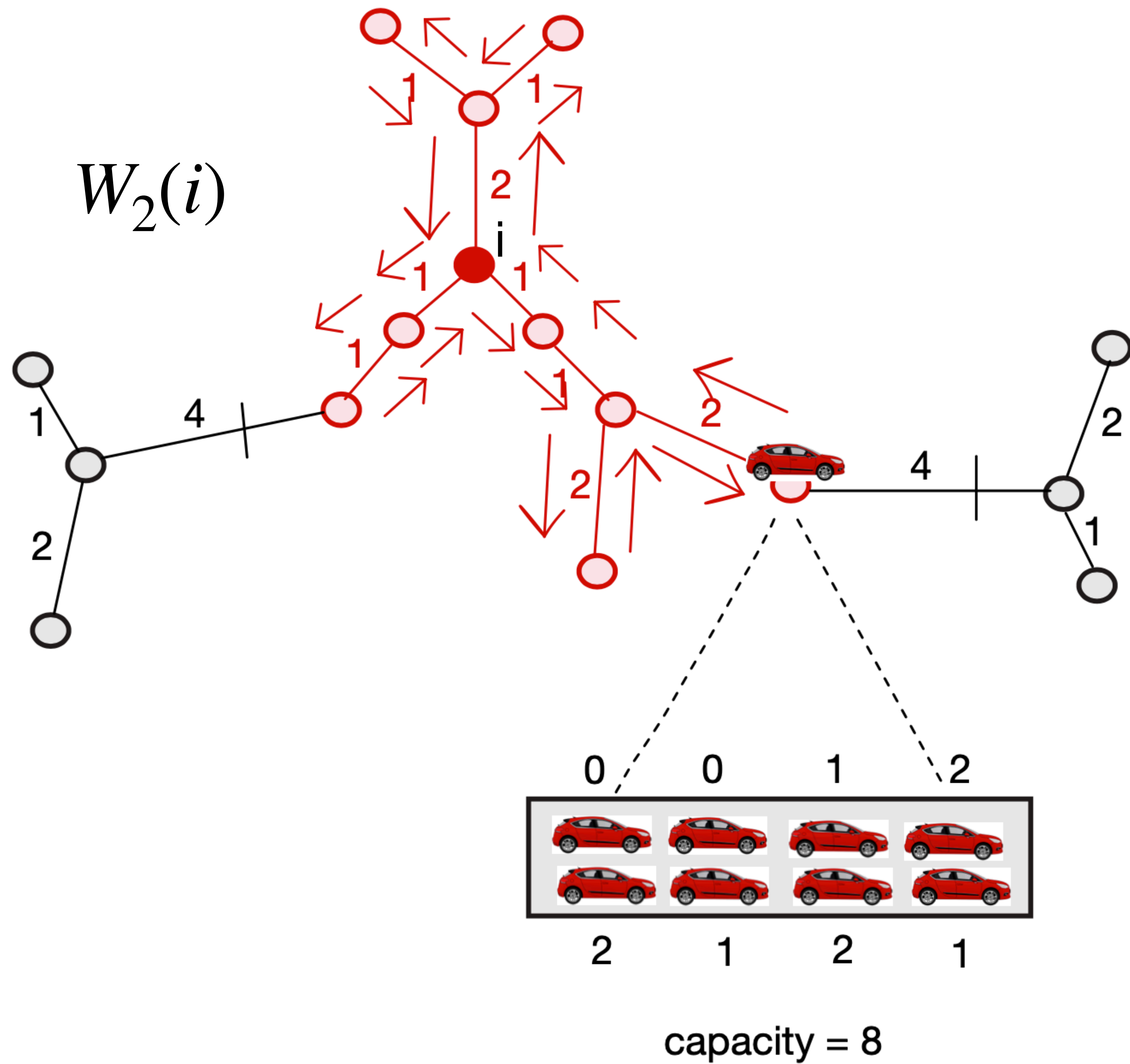


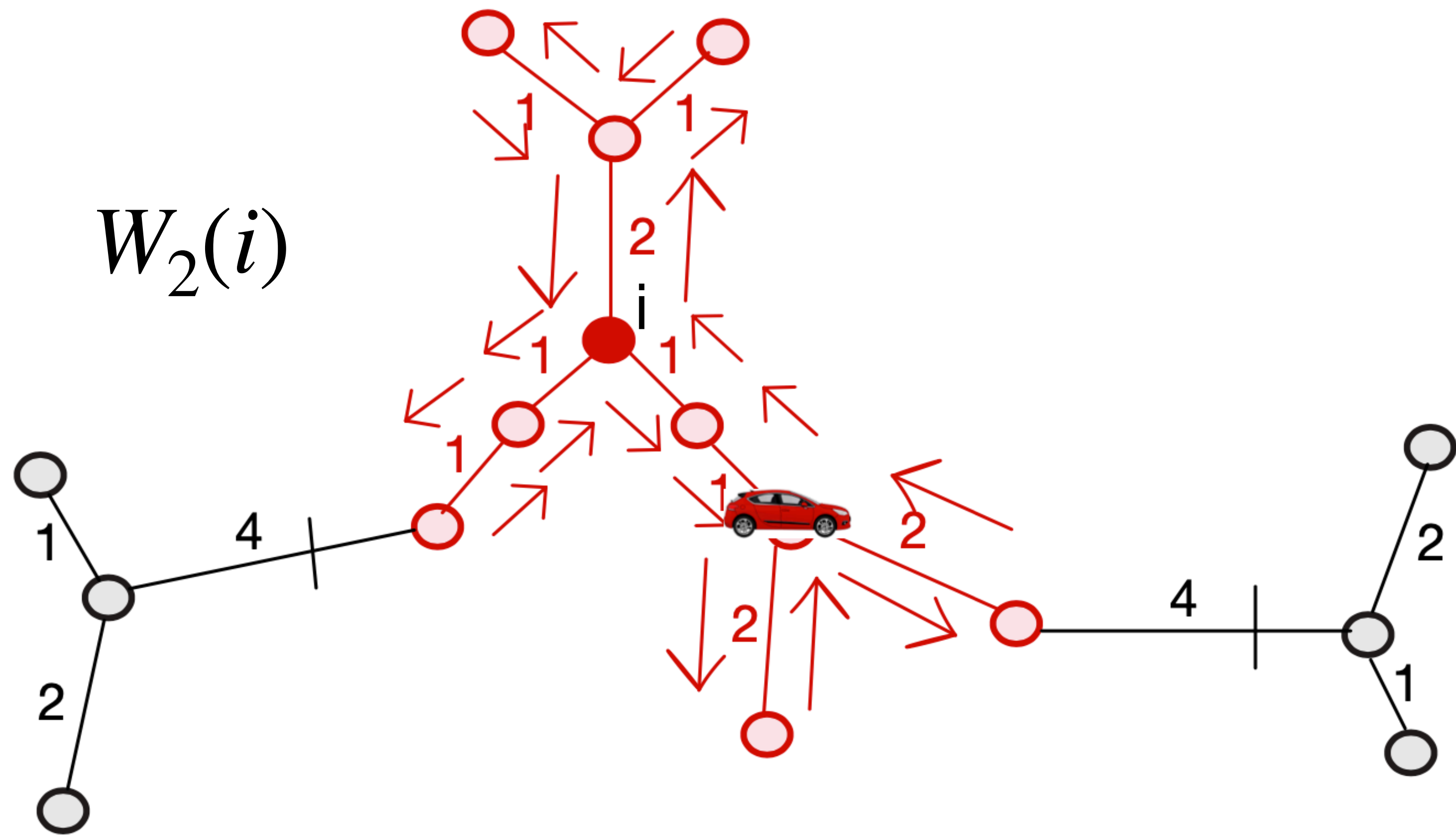






capacity = 8





Algorithm A

- Conceptually...
 - Each car is focused on driving its own route
 - Each car parks whenever it can
 - **Lower-level cars** get **priority** when parking

Algorithm A

- Execution of Algorithm A
 - Car arrives at vertex i and drives $W_0(i)$
 - If full, kicks out level 2 car, which starts driving $W_2(j)$
 - If full, kicks out level 3 car, which starts driving $W_3(k)$
 - If full, kicks out level 5 car, which starts driving $W_5(l)$
 - If unfull, level 5 car parks, **completing the time step.**

Algorithm B

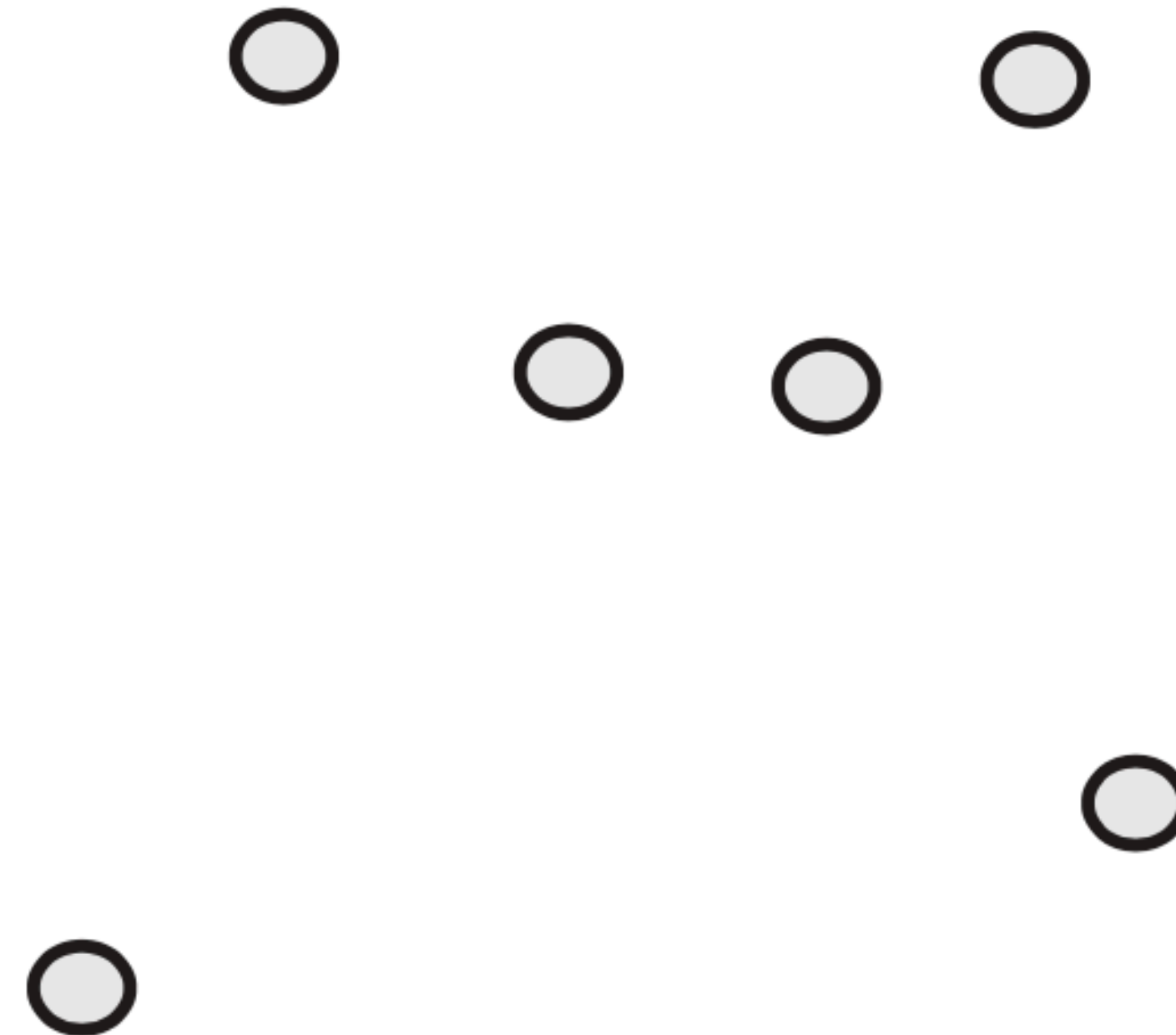
Itinerant Car

General Metrics

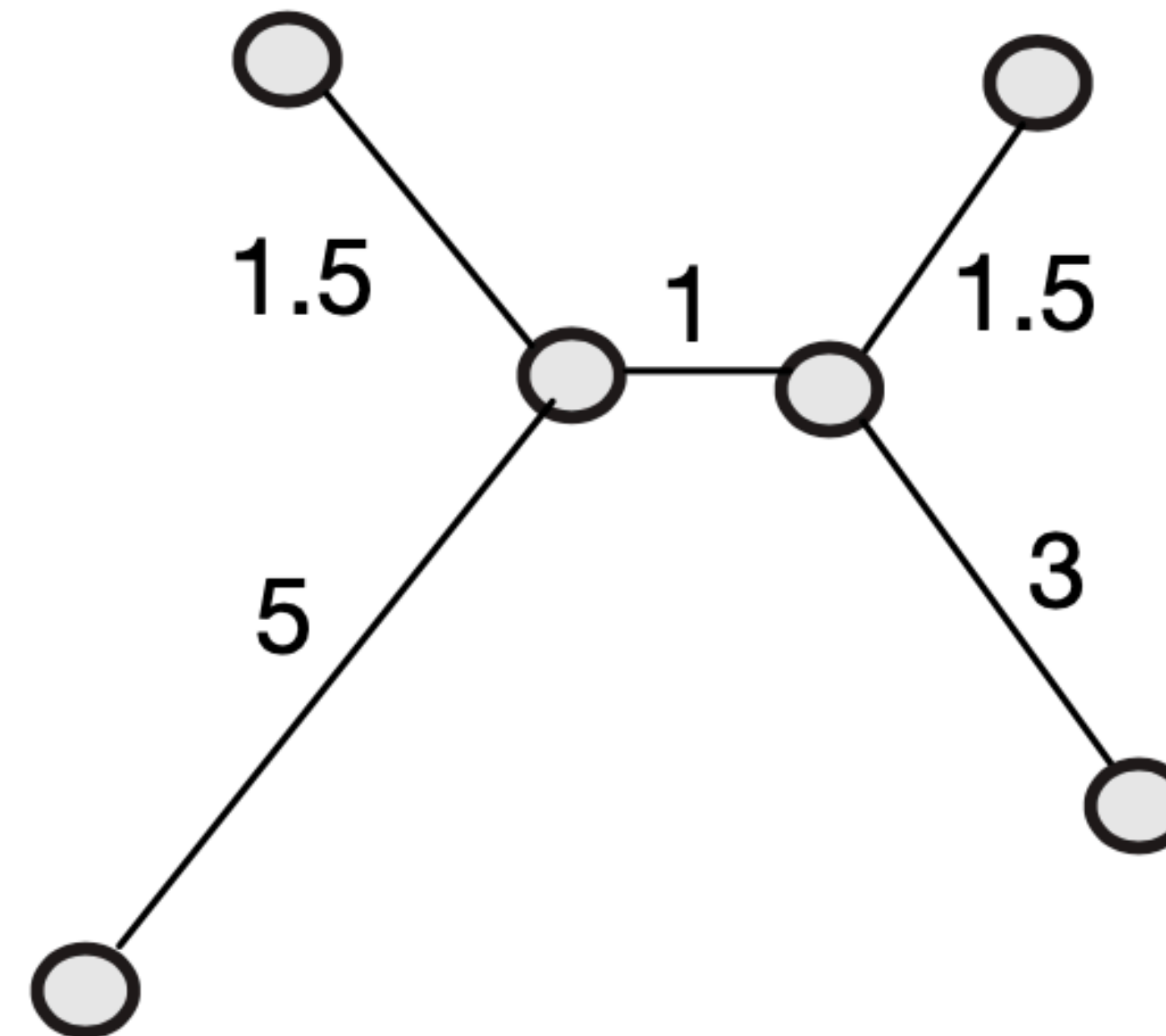
Algorithm B

- Algorithm B simulates Algorithm A.
- **Setting:** Metric space $M = (X, d)$, m garages with capacities $c_1, c_2, \dots, c_m$
 - Build clique K_m on the garages with edge lengths $d(u, v)$.
 - Find MST S
 - Round all edge lengths **up** to next **integer power of two**.

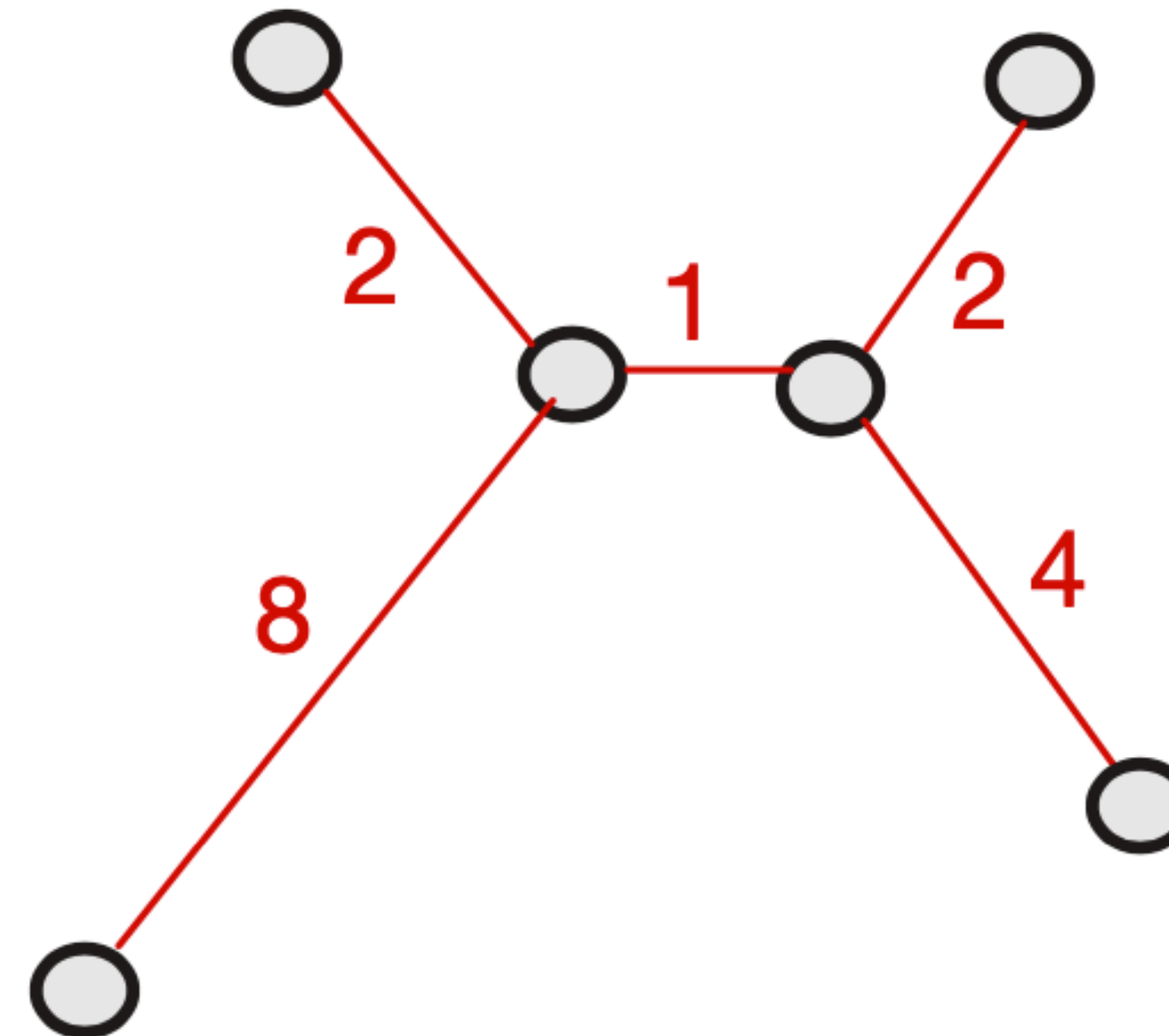
Algorithm B



Algorithm B

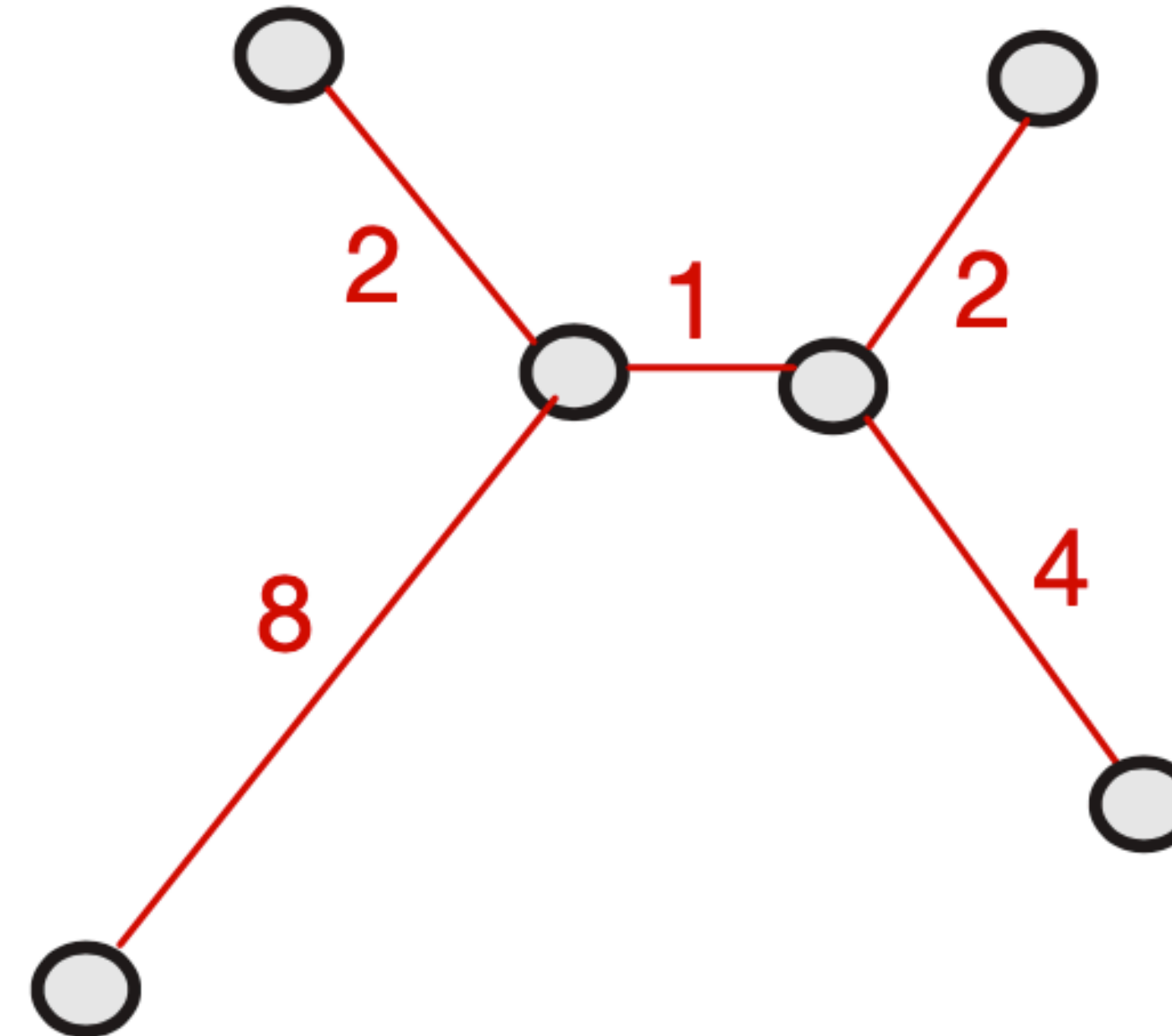


Algorithm B



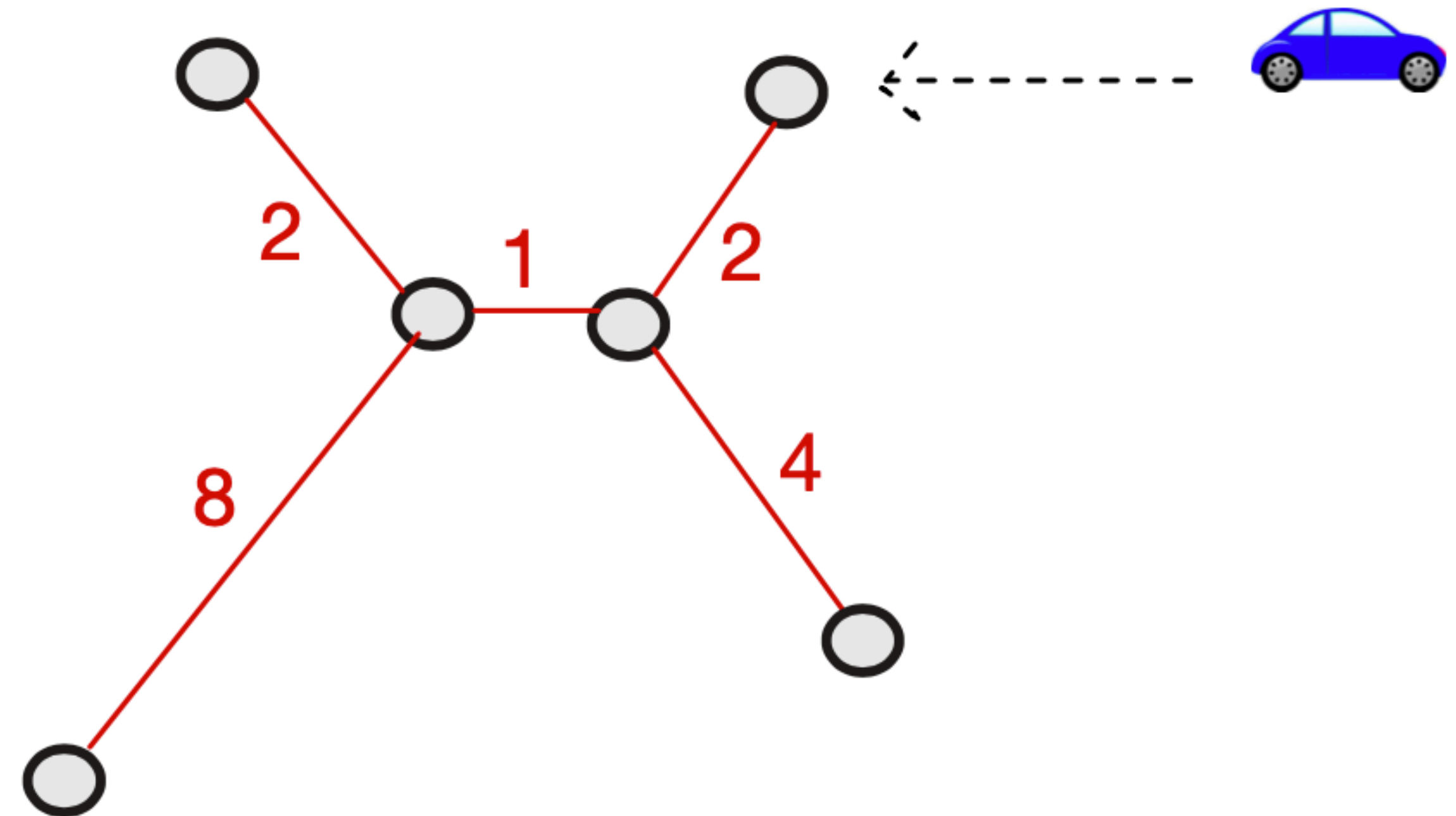
Algorithm B

- **Online Input:** Car arrives at $x \in X$



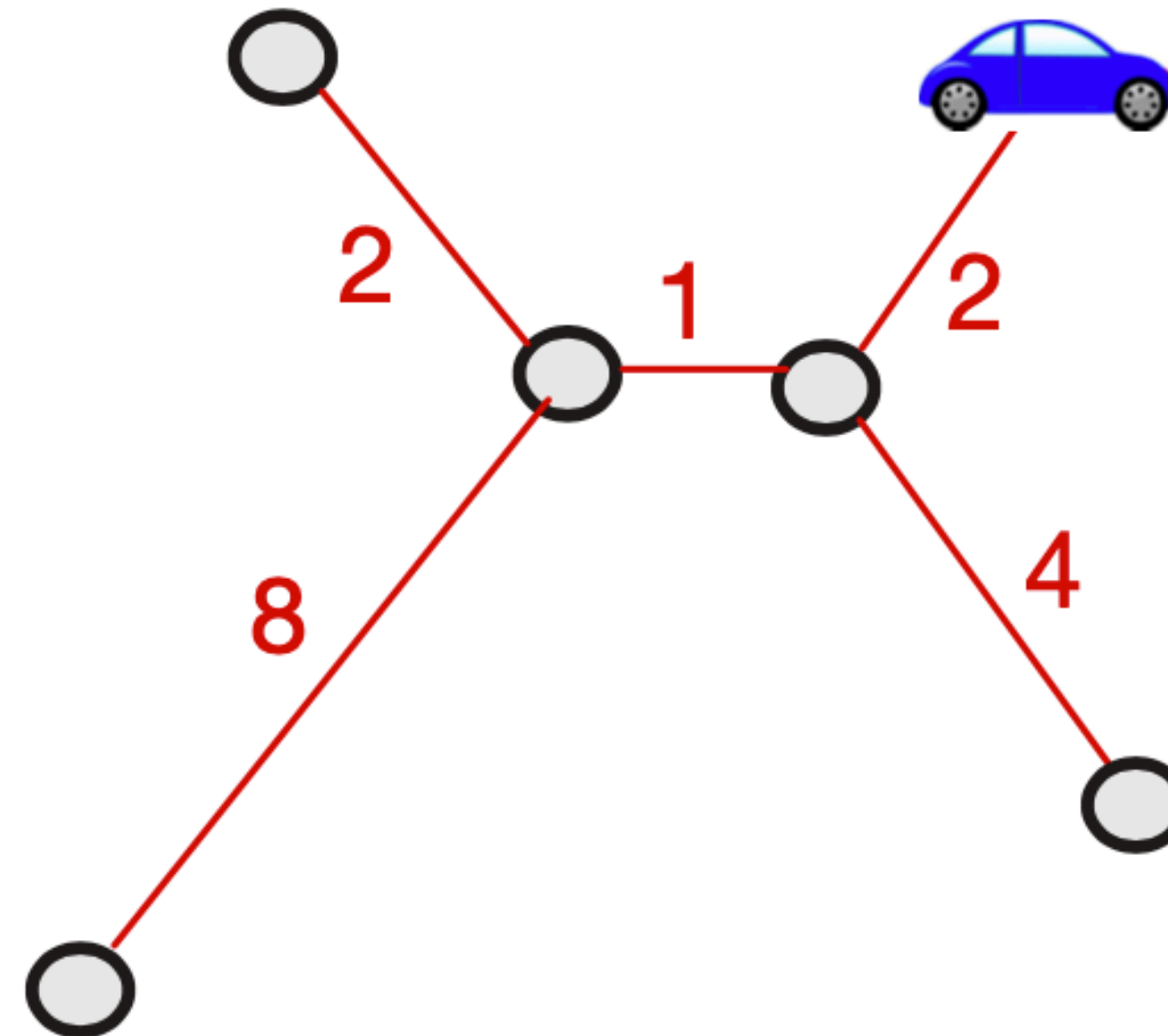
Algorithm B

- **Online Input:** Car arrives at $x \in X$
 - Moves car to nearest garage



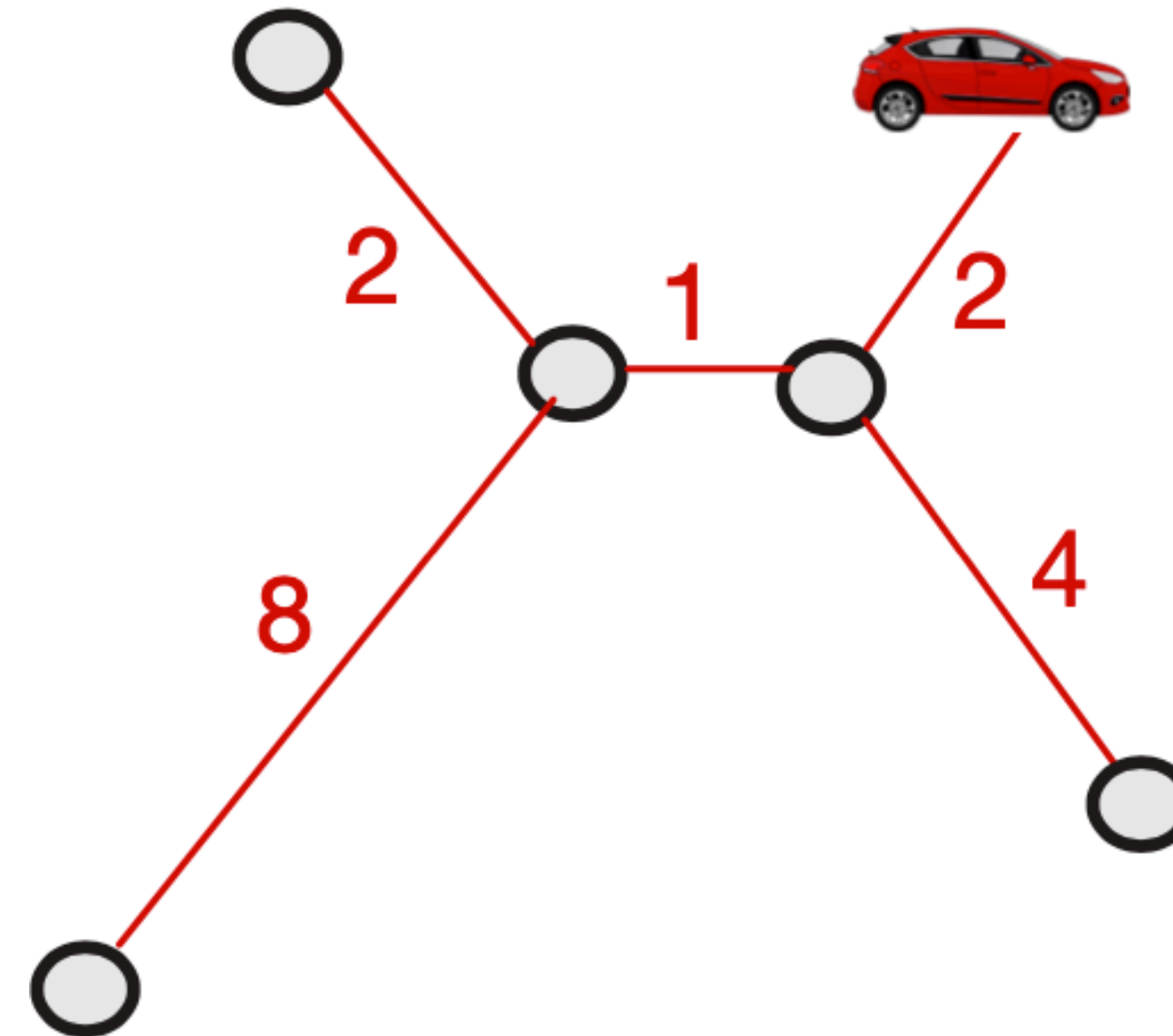
Algorithm B

- **Online Input:** Car arrives at $x \in X$
 - Moves car to nearest garage



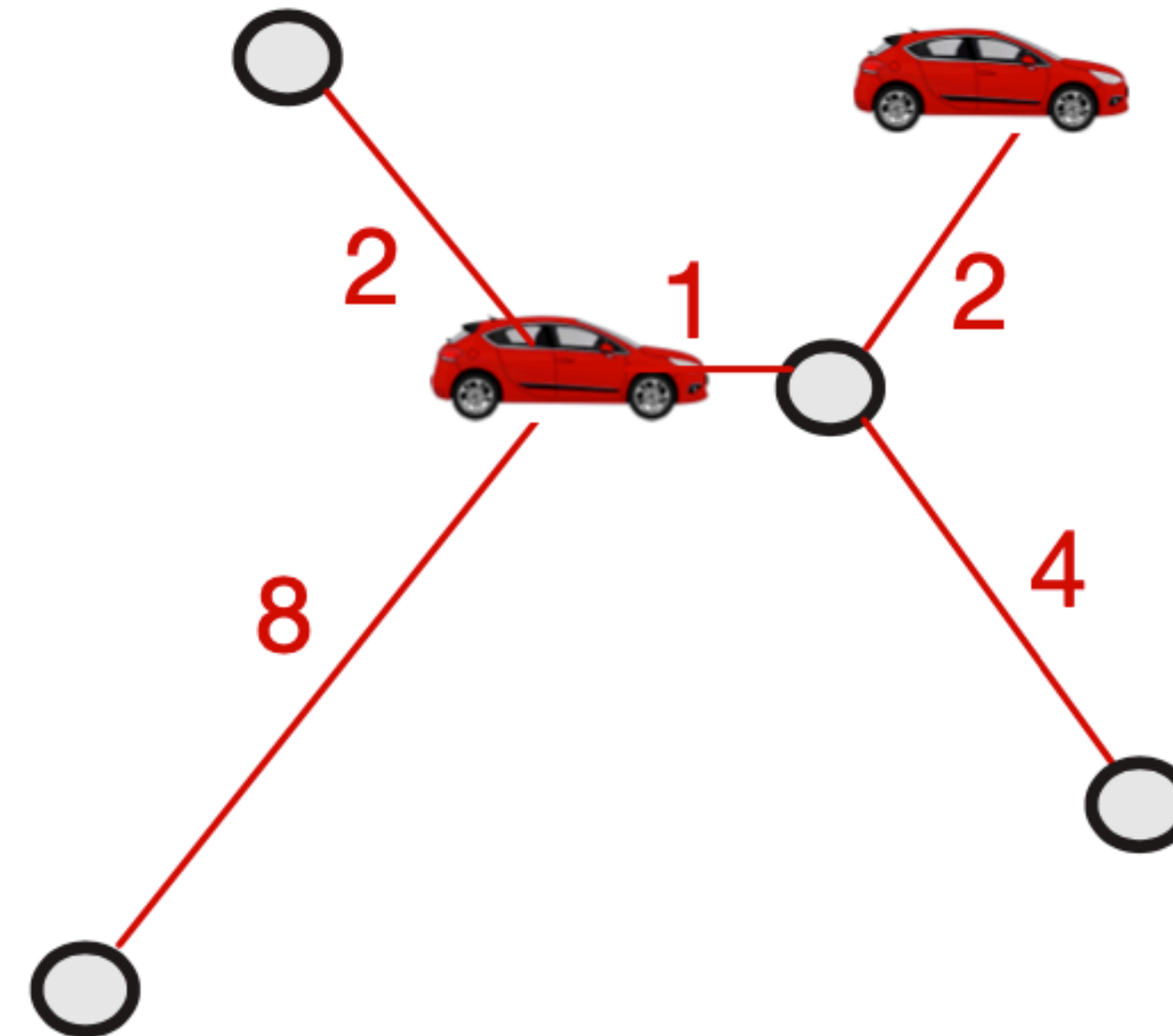
Algorithm B

- **Online Input:** Car arrives at $x \in X$
 - Moves car to nearest garage
 - Simulates **Algorithm A**



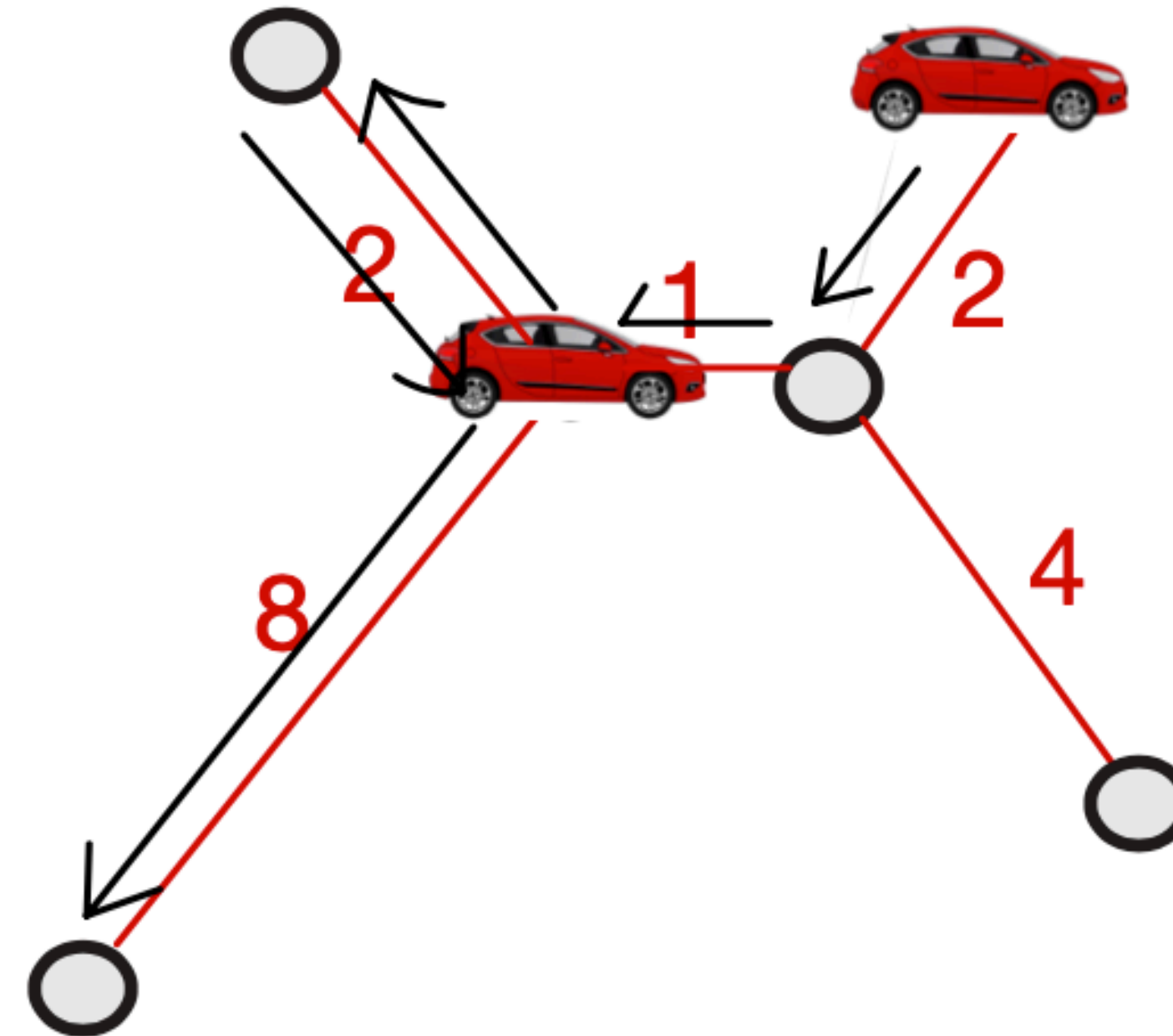
Algorithm B

- **Online Input:** Car arrives at $x \in X$
 - Moves car to nearest garage
 - Simulates **Algorithm A**



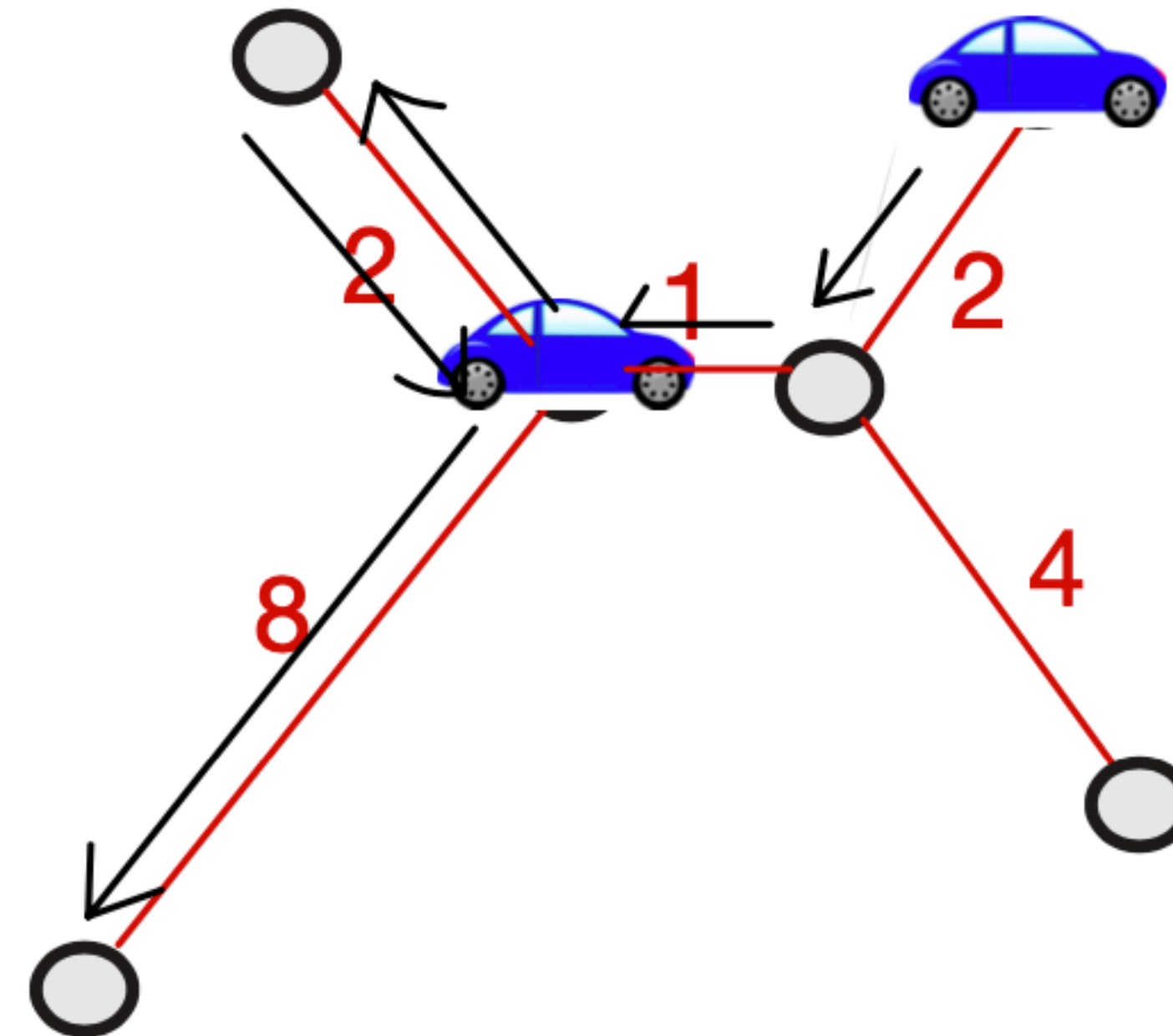
Algorithm B

- **Online Input:** Car arrives at $x \in X$
 - Moves car to nearest garage
 - Simulates **Algorithm A**



Algorithm B

- **Online Input:** Car arrives at $x \in X$
 - Moves car to nearest garage
 - Simulates **Algorithm A**
 - Moves cars identically



Algorithm C

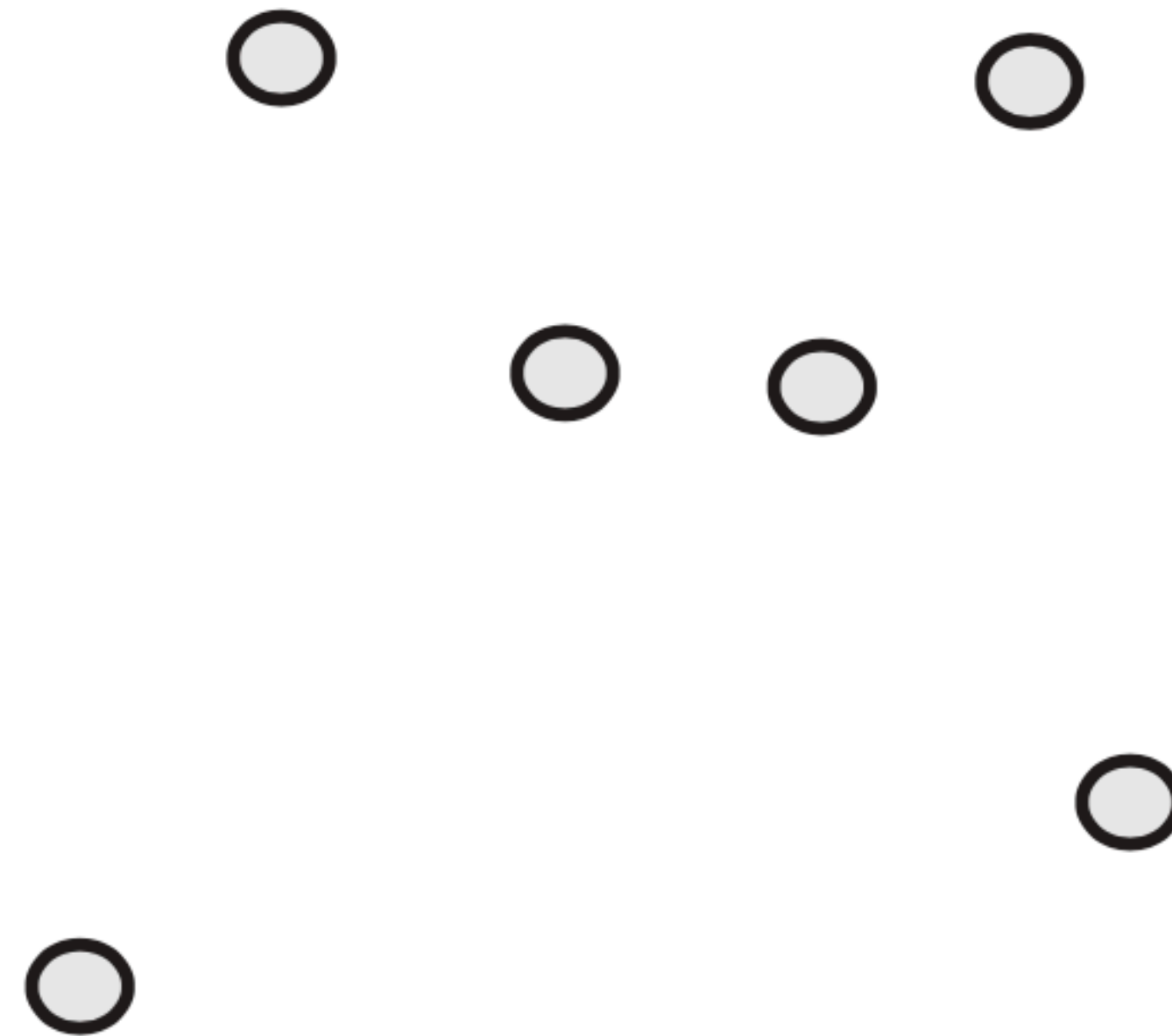
Online Transportation

General Metrics

Algorithm C

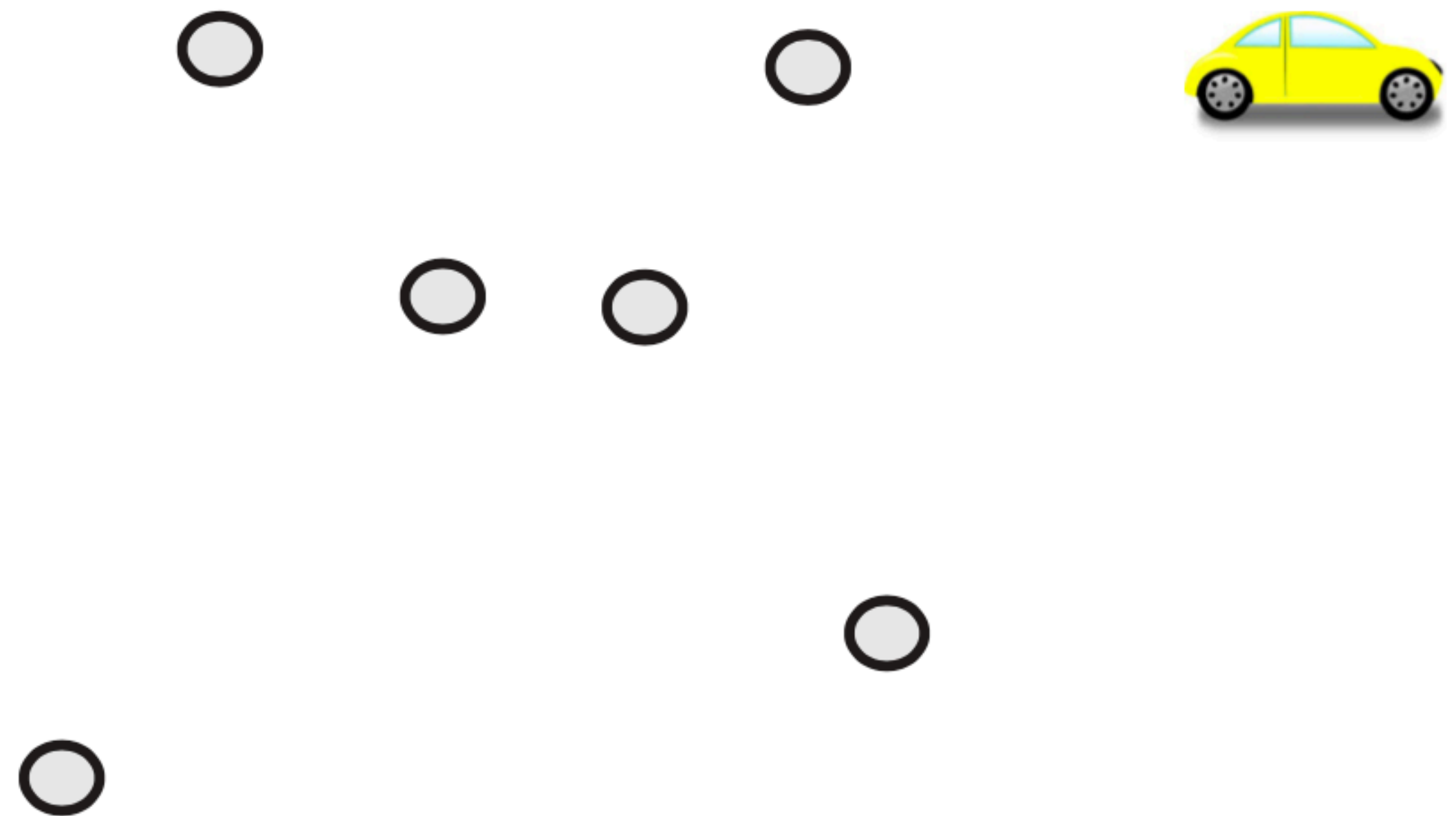
- Algorithm C simulates Algorithm B.
- **Setting:** Metric space $M = (X, d)$, m garages with capacities $c_1, c_2, \dots, c_m$
- **Online Input:**
 - Car arrives at location $x \in X$
 - Must be assigned to garage $i \in [m]$
 - Continues until all garages full
- **Objective: Minimize** total distance cars travel

Algorithm C



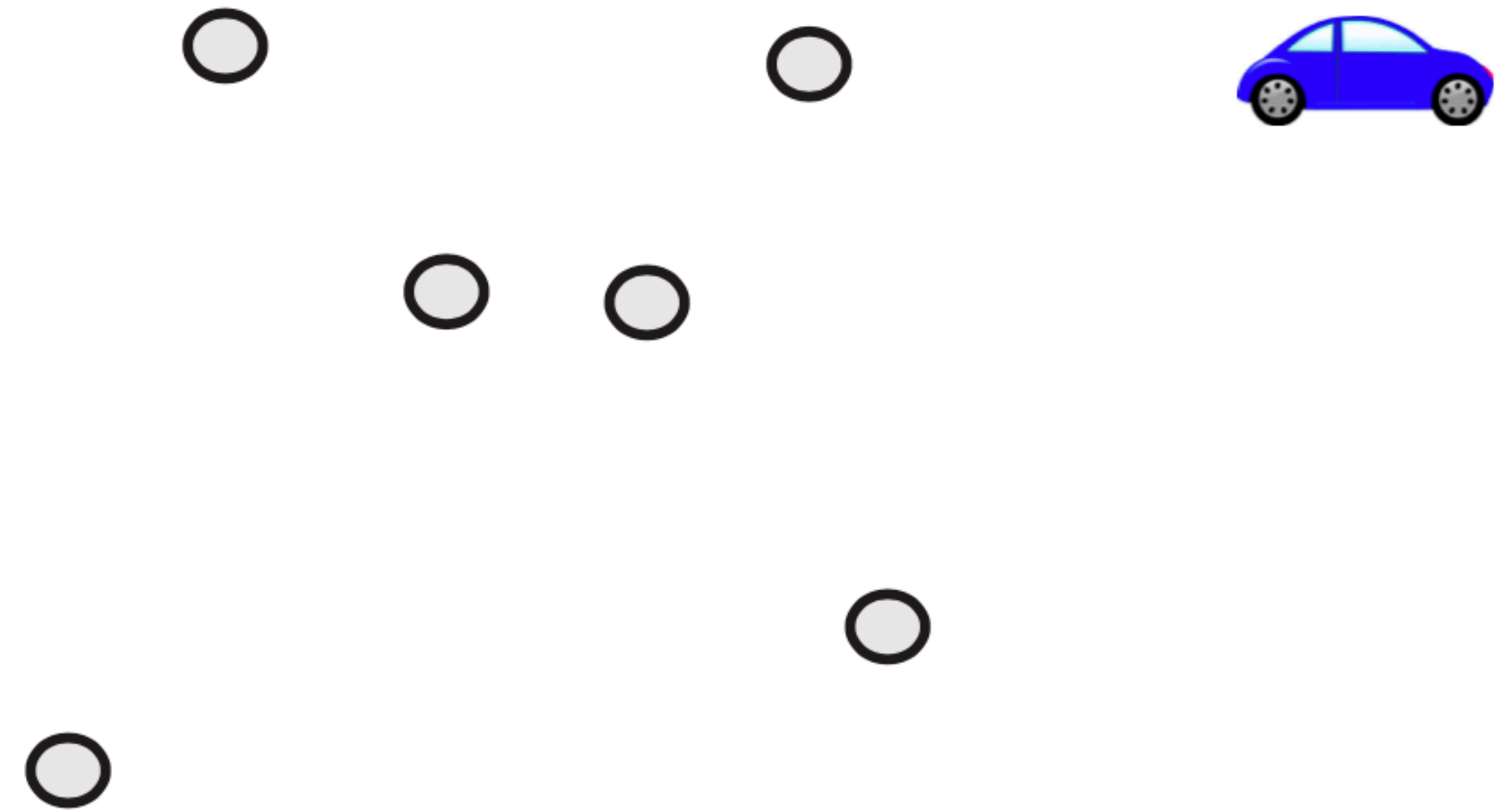
Algorithm C

- **Online Input:** Car arrives at $x \in X$



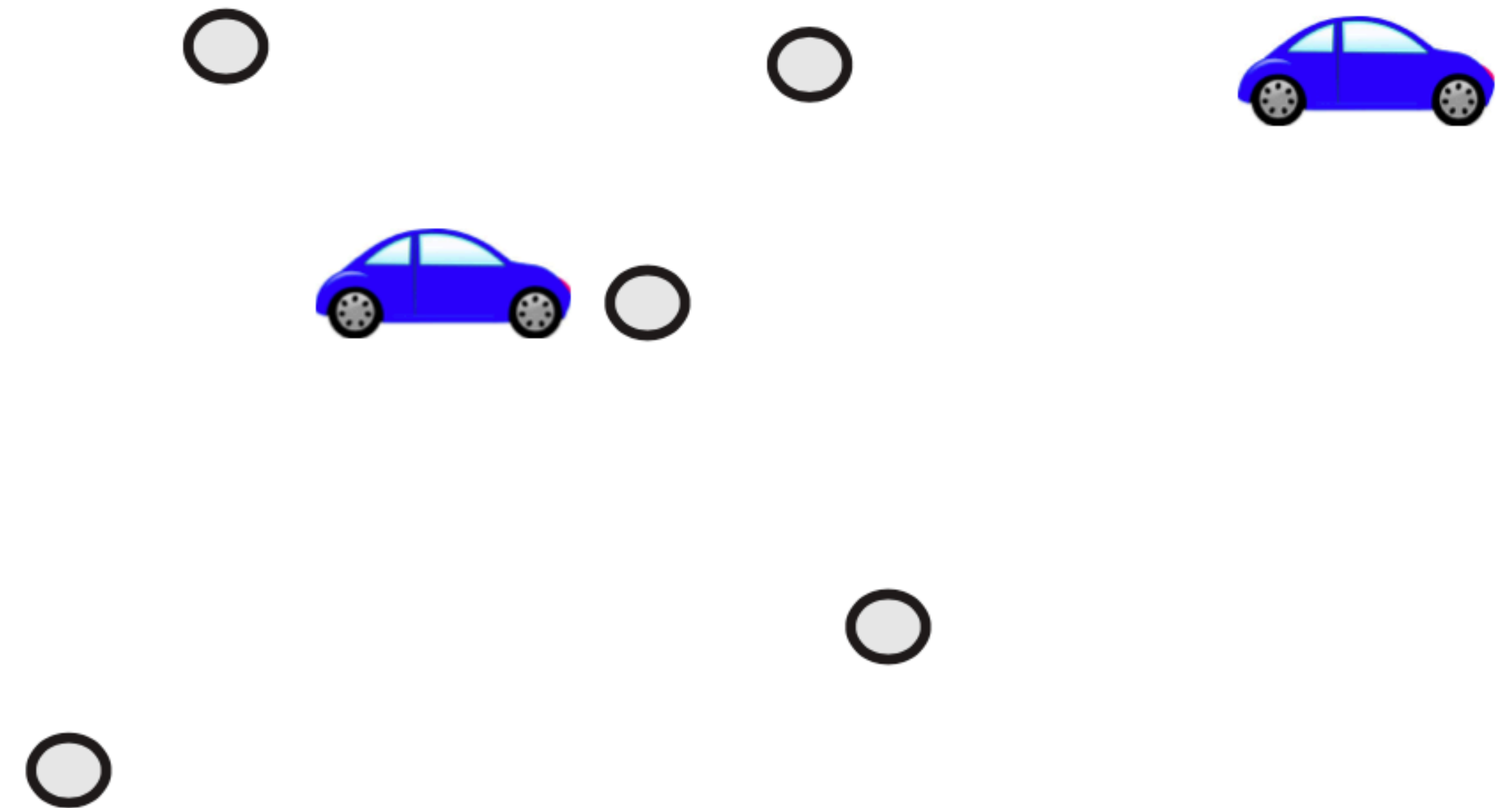
Algorithm C

- **Online Input:** Car arrives at $x \in X$
 - Simulates Algorithm B



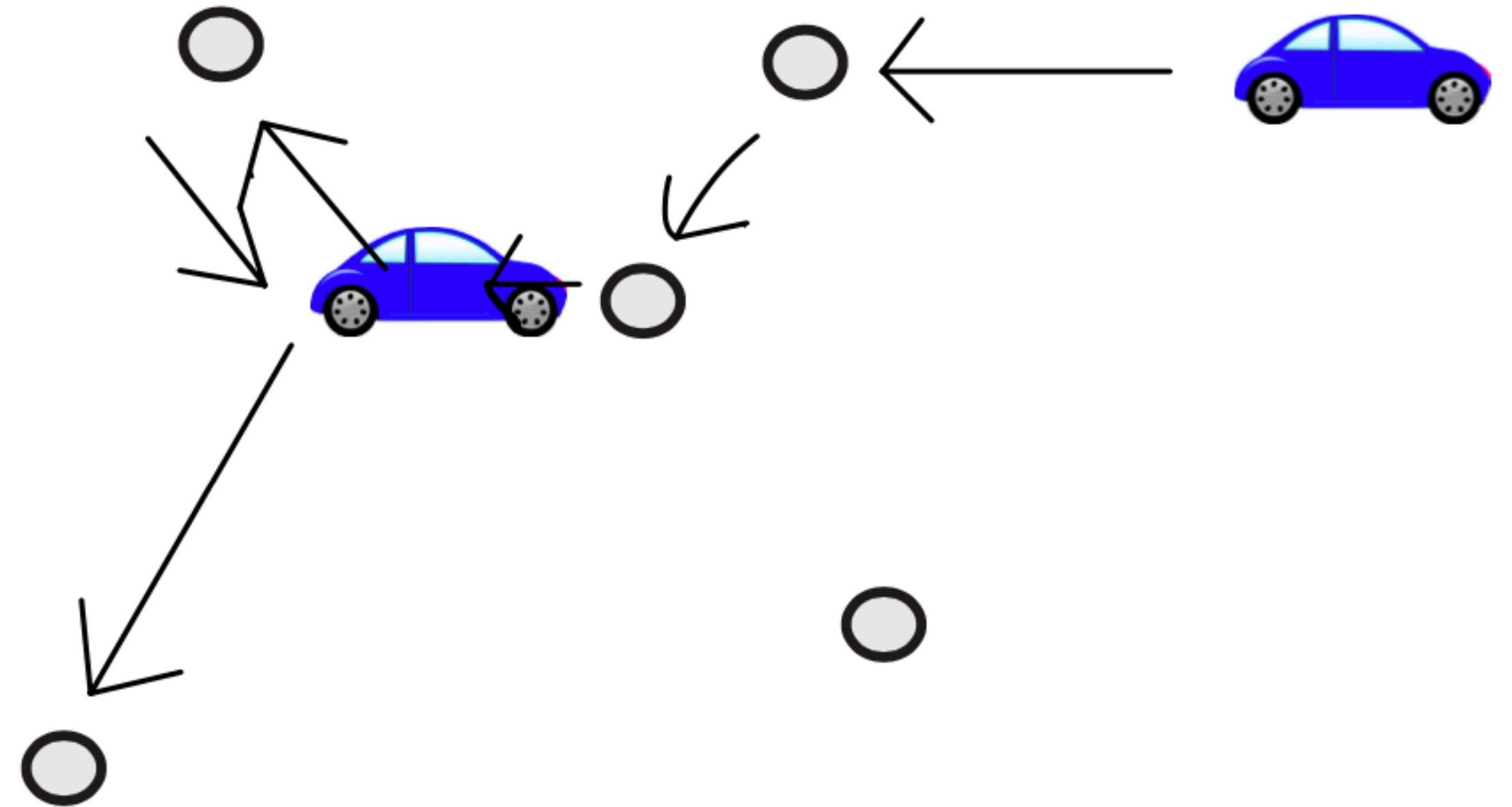
Algorithm C

- **Online Input:** Car arrives at $x \in X$
 - Simulates Algorithm B



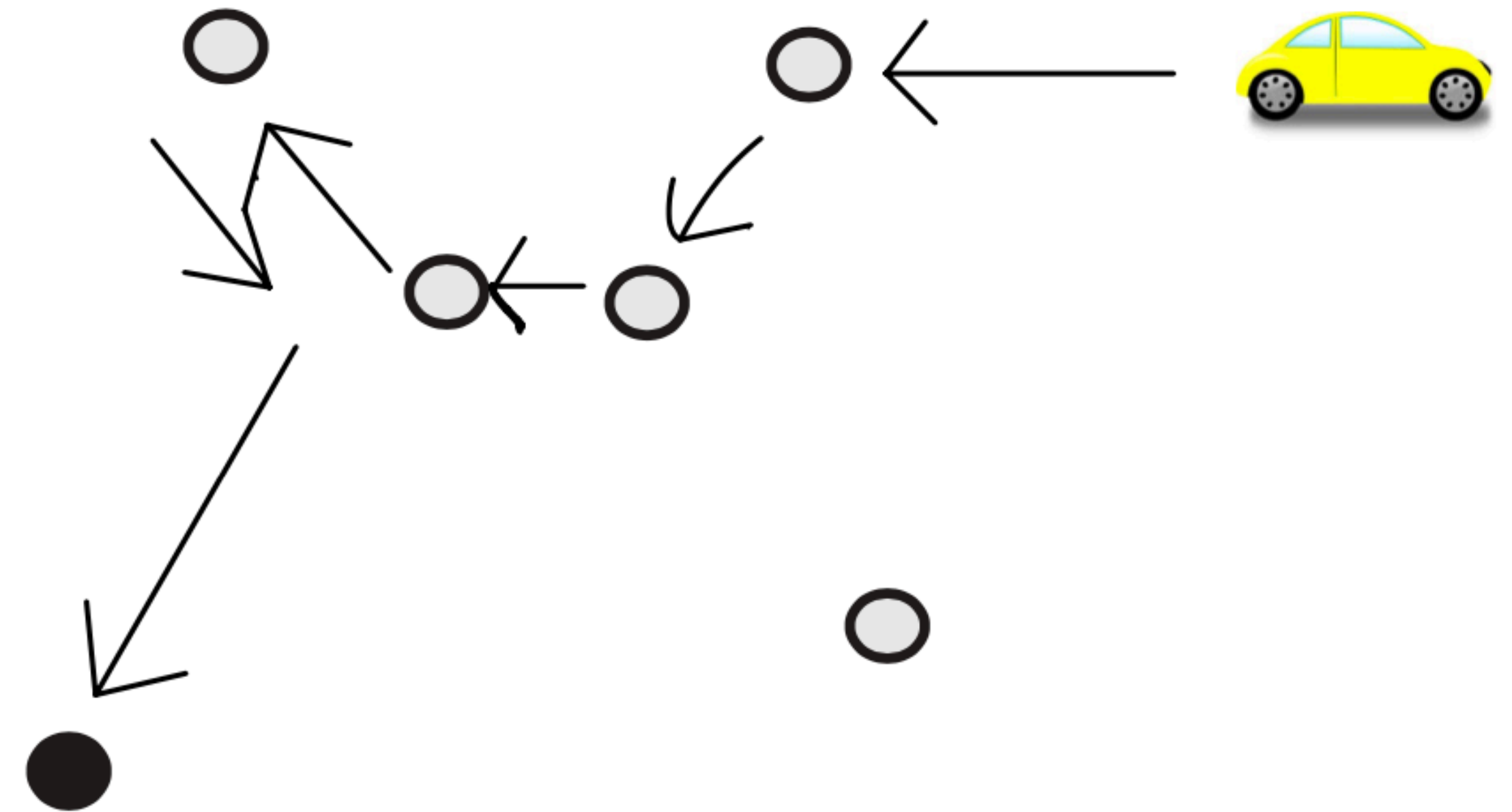
Algorithm C

- **Online Input:** Car arrives at $x \in X$
 - Simulates **Algorithm B**



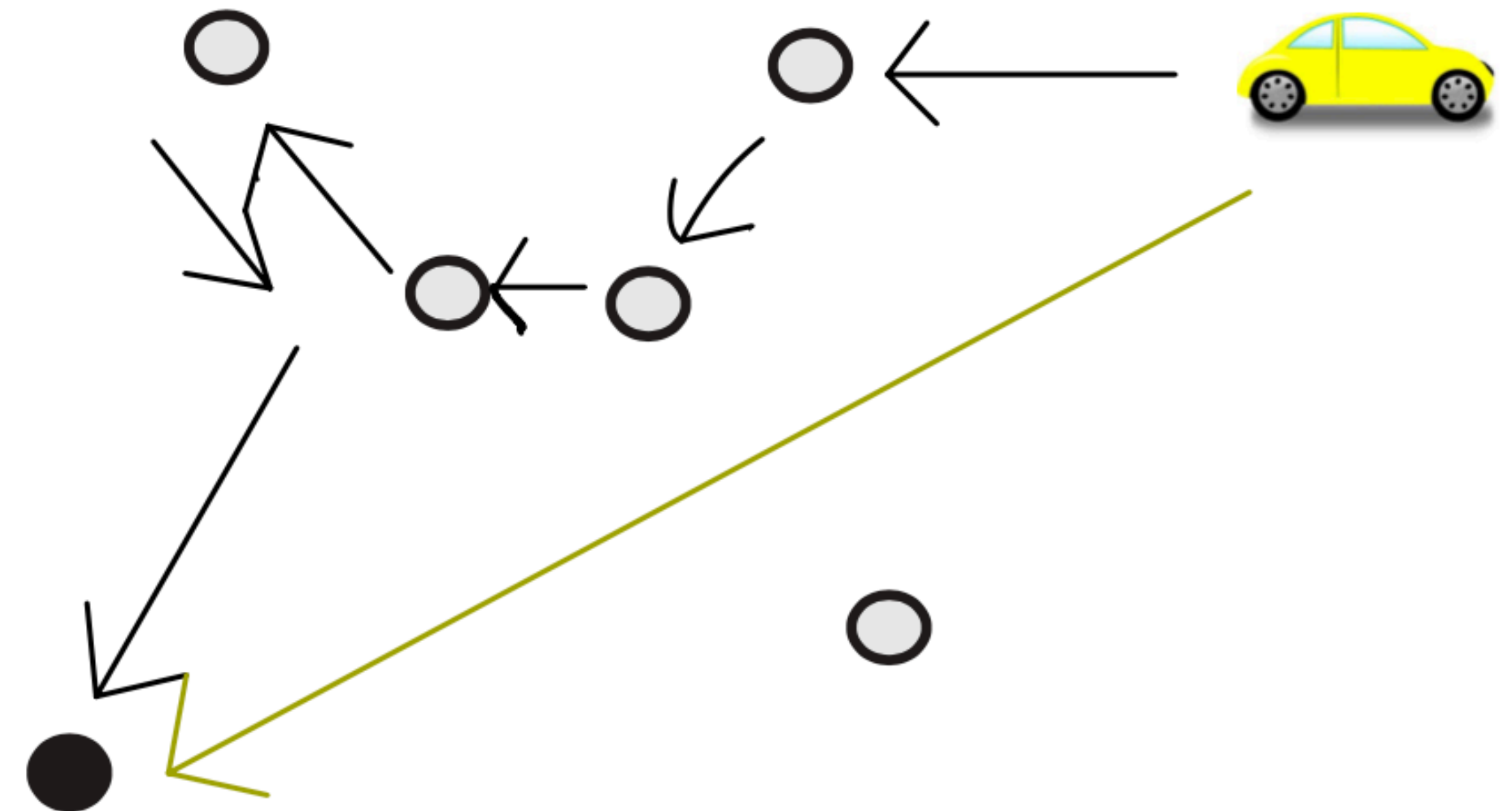
Algorithm C

- **Online Input:** Car arrives at $x \in X$
 - Simulates **Algorithm B**

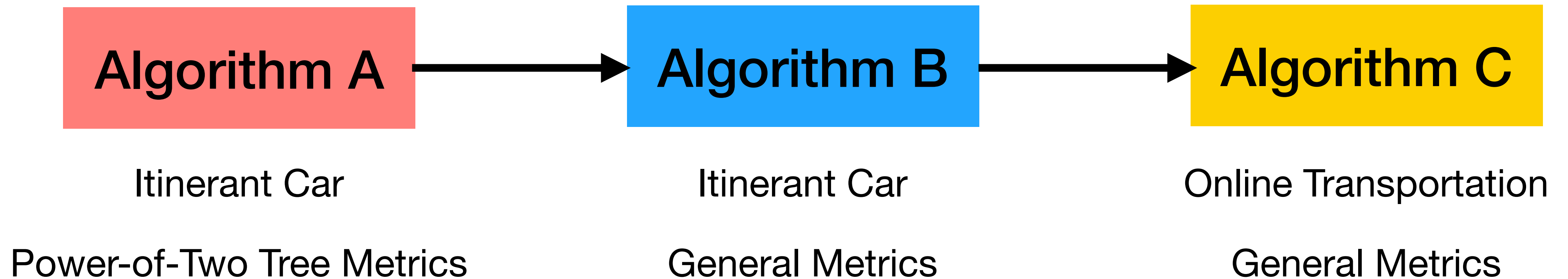


Algorithm C

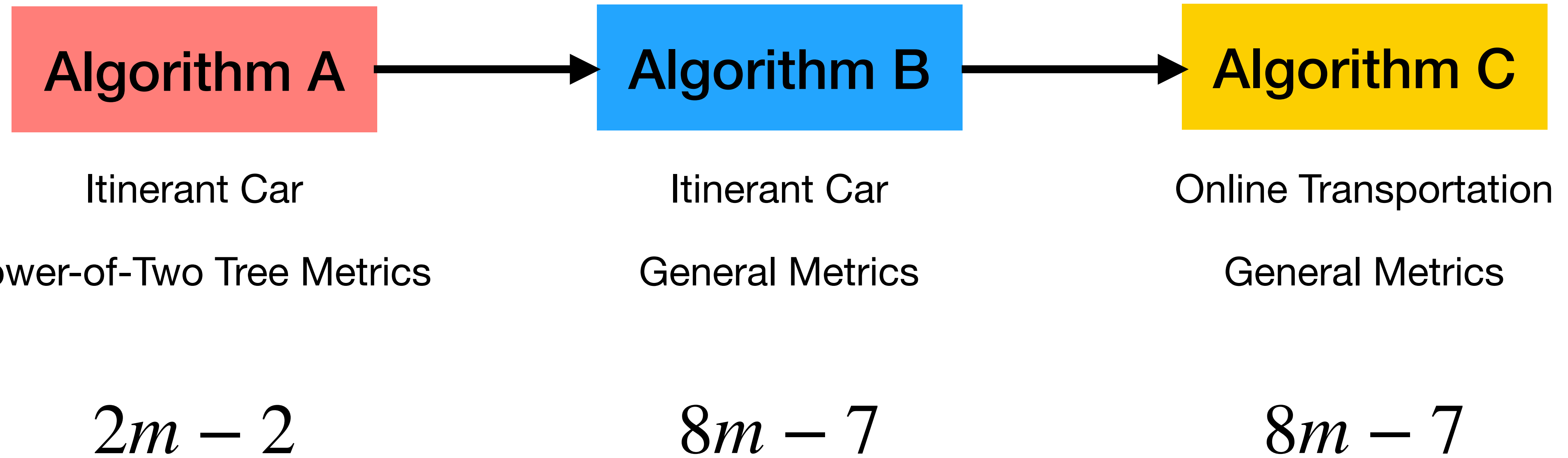
- **Online Input:** Car arrives at $x \in X$
 - Simulates **Algorithm B**
 - Moves car to **final garage** on chain



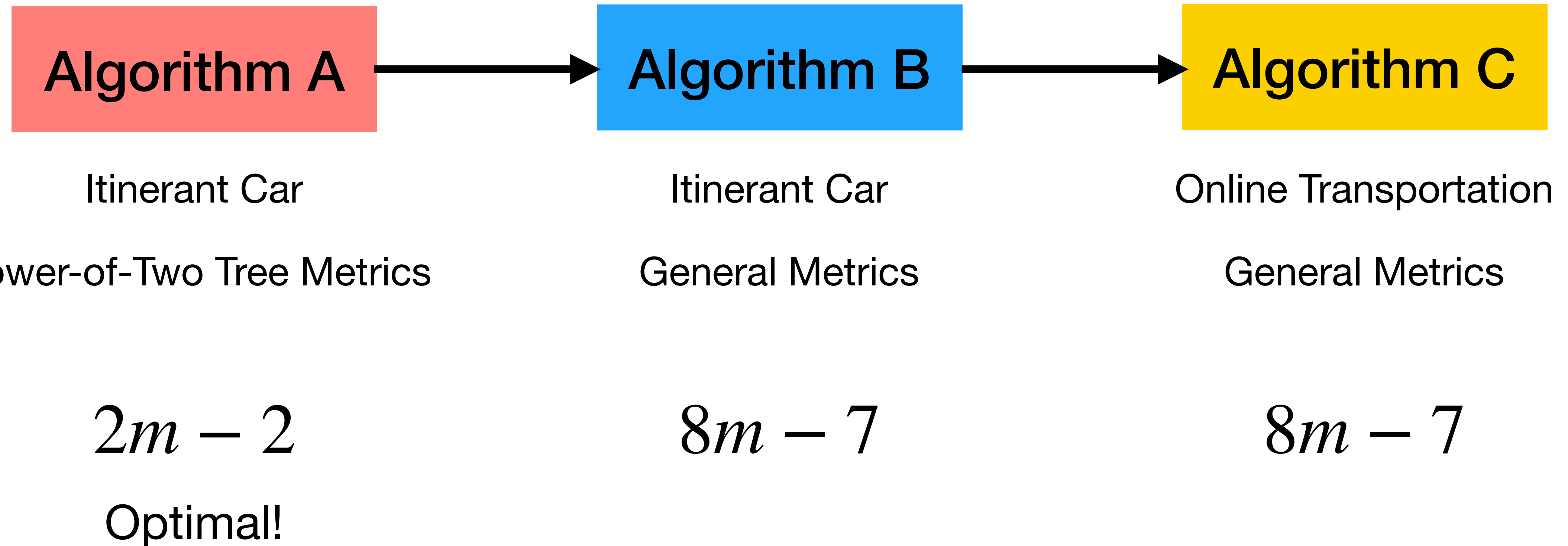
Analysis: Brief Overview



Analysis: Brief Overview



Analysis: Brief Overview



Analysis: Brief Overview



Itinerant Car

Itinerant Car

Online Transportation

Power-of-Two Tree Metrics

General Metrics

General Metrics

CR: $2m - 2$

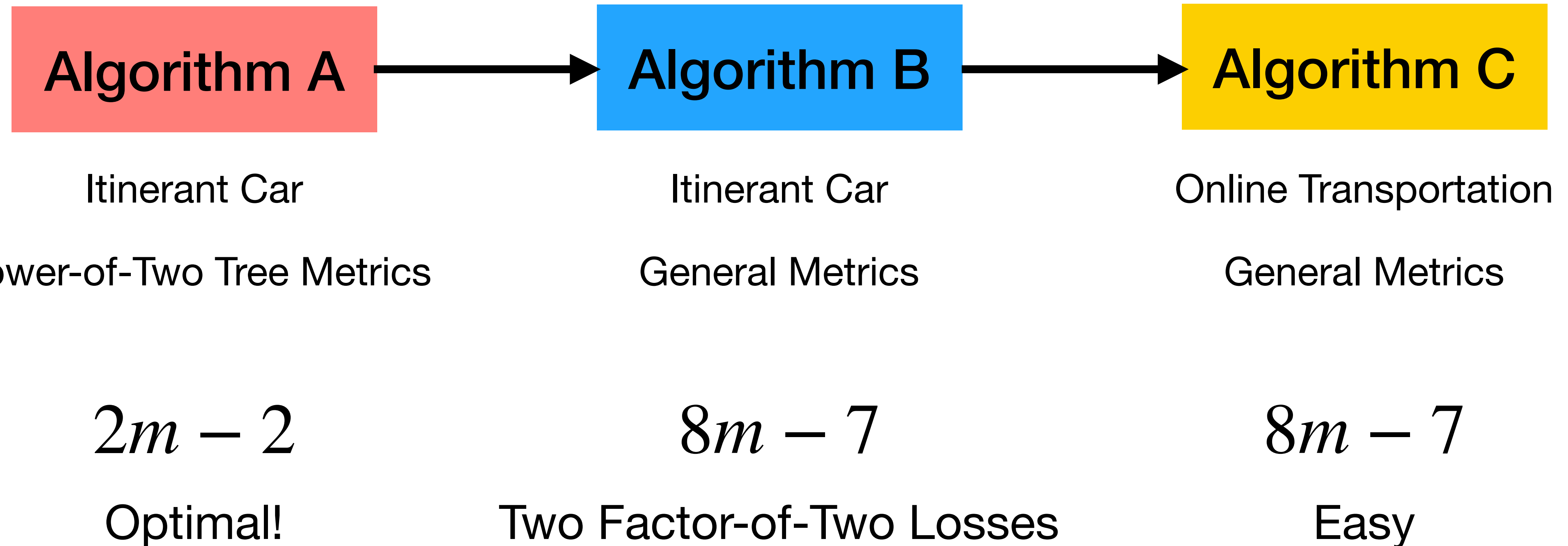
$8m - 7$

$8m - 7$

Optimal!

Two Factor-of-Two Losses

Analysis: Brief Overview



Conclusion

- **Online Transportation Problem**
 - **Conj** (**Kalyanasundaram, Pruhs**): There is a **$(2m-1)$** -competitive algorithm.
 - **Thm** (**Harada, Itoh**): There is a **$(8m-5)$** -competitive algorithm.
 - **Thm**: There is a **$(8m-7)$** -competitive “simple” algorithm + analysis.
 - **Hope**: Our simplifications help the field get down to $2m-1$.
 - **Where to Start?** Fix those two factor-of-two losses in **Algorithm B**!

Thank you!

Questions?

