

Esercitazione
18 marzo 2025

Algoritmi greedy

problema 1

Esercizio svolto #3 (pag. 187)

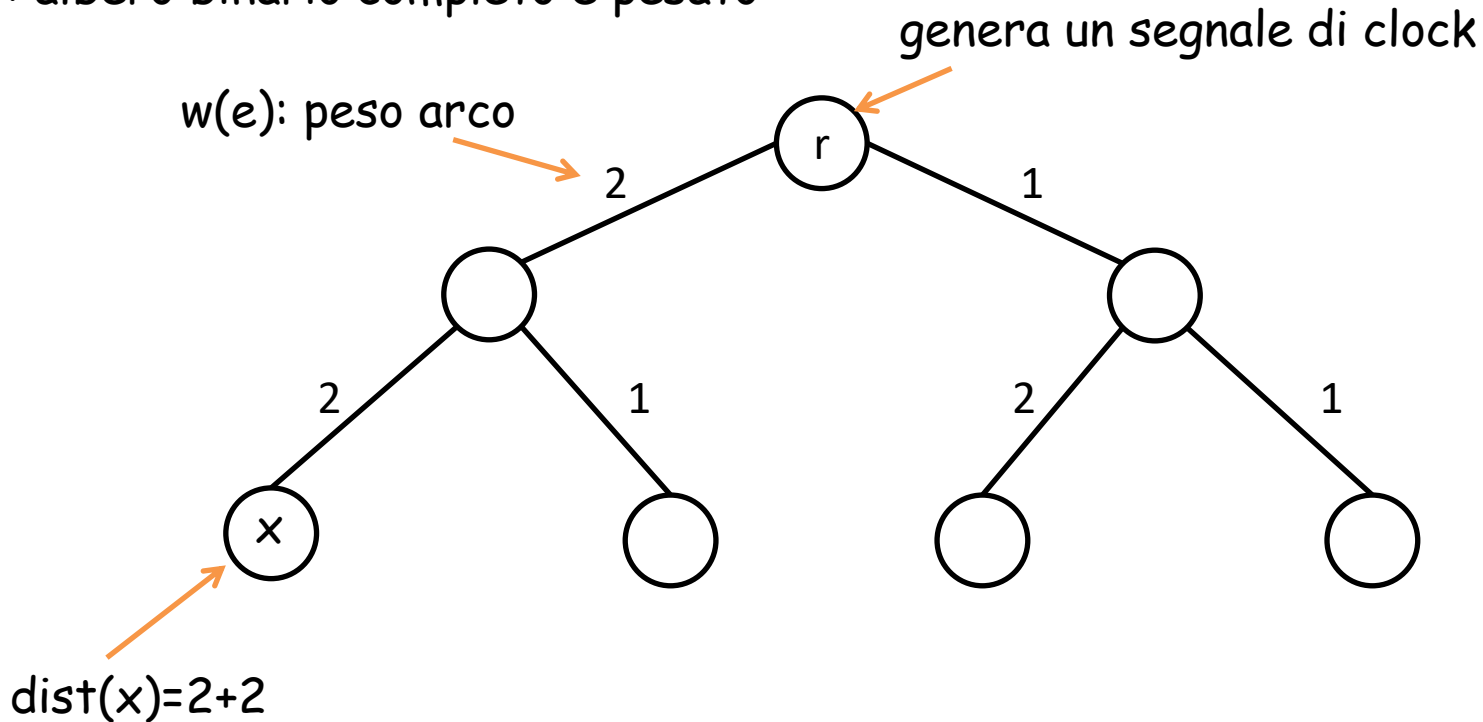
Sia $G = (V, E)$ un grafo con pesi positivi distinti sugli archi ed $e \in E$ un arco di G . Progettare un algoritmo lineare in grado di determinare se esiste un MST di G che contiene l'arco e .

problema 2

Esercizio #24 (pag. 200)

sincronizzazione di circuiti

T: albero binario completo e pesato



tempo a cui arriva il segnale

goal: sincronizzare le foglie: metterle tutte alla stessa distanza

come: posso incrementare i pesi degli archi

misura (da minimizzare): peso totale dell'albero risultante (min somma incrementi)

input:

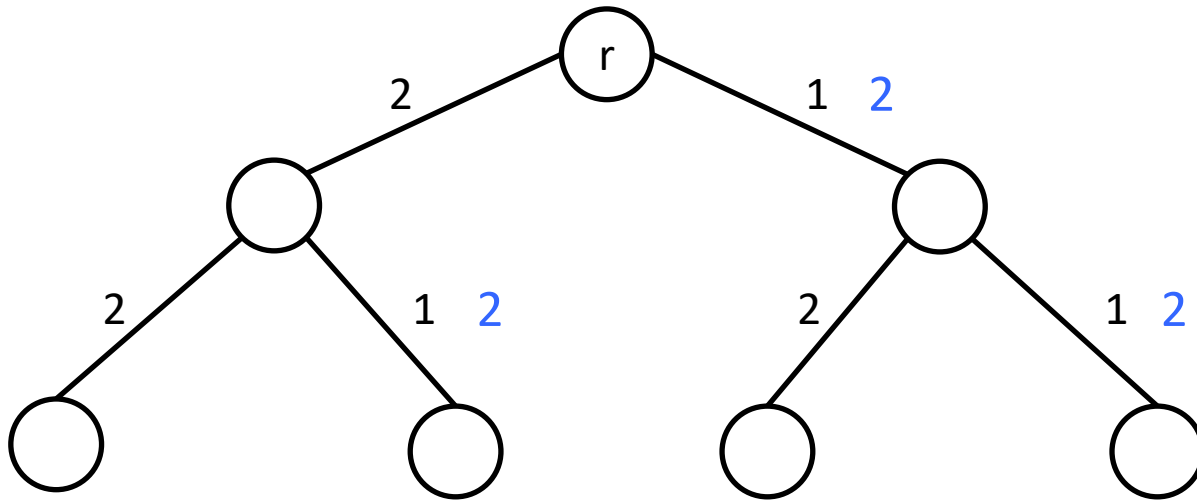
albero binario completo e pesato T

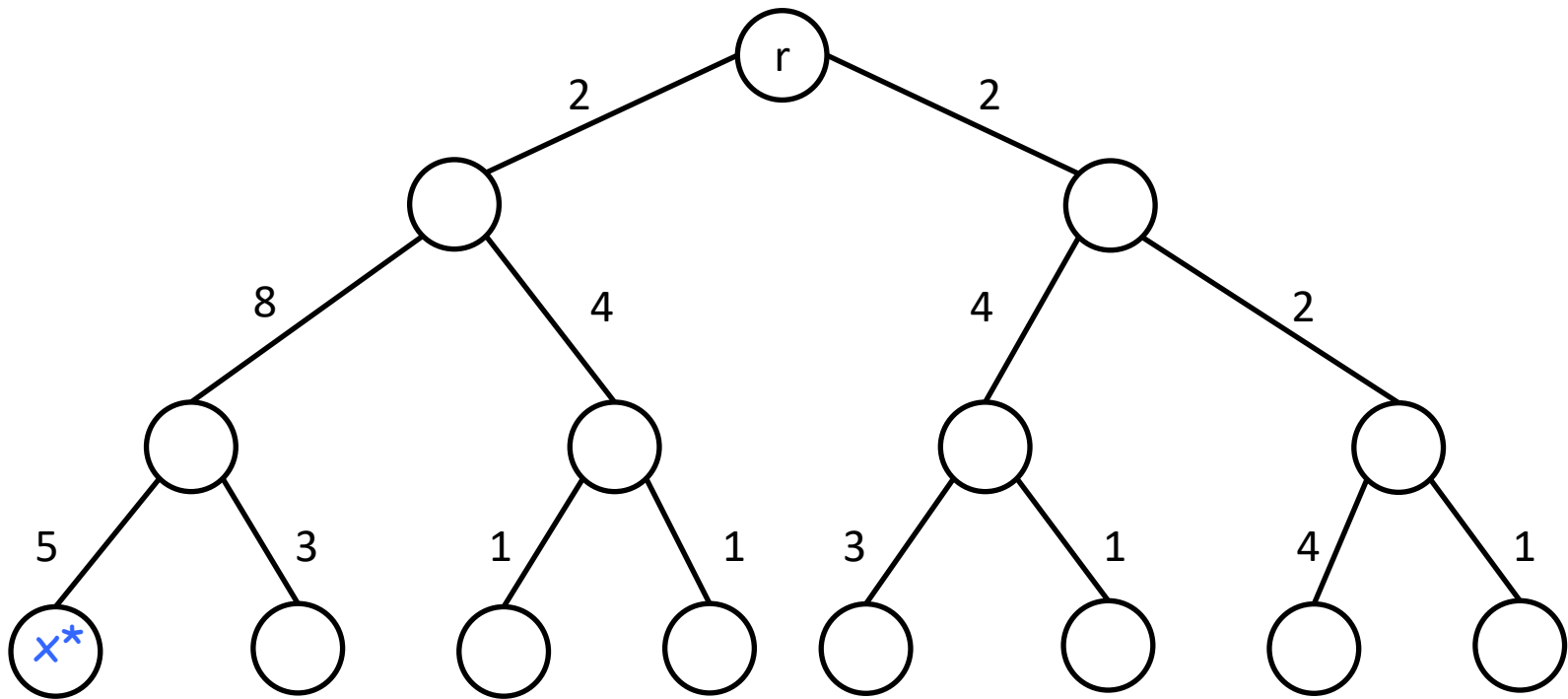
soluzione ammissibile:

nuova pesatura w' per T con $w'(e) \geq w(e)$ tale che tutte le foglie rispetto a w' hanno la stessa distanza dalla radice

misura (da minimizzare):

$$w'(T) = \sum_e w'(e)$$





x^* : foglia più lontana a distanza $L=15$

idea: mettere tutte le foglie a distanza $L=15$

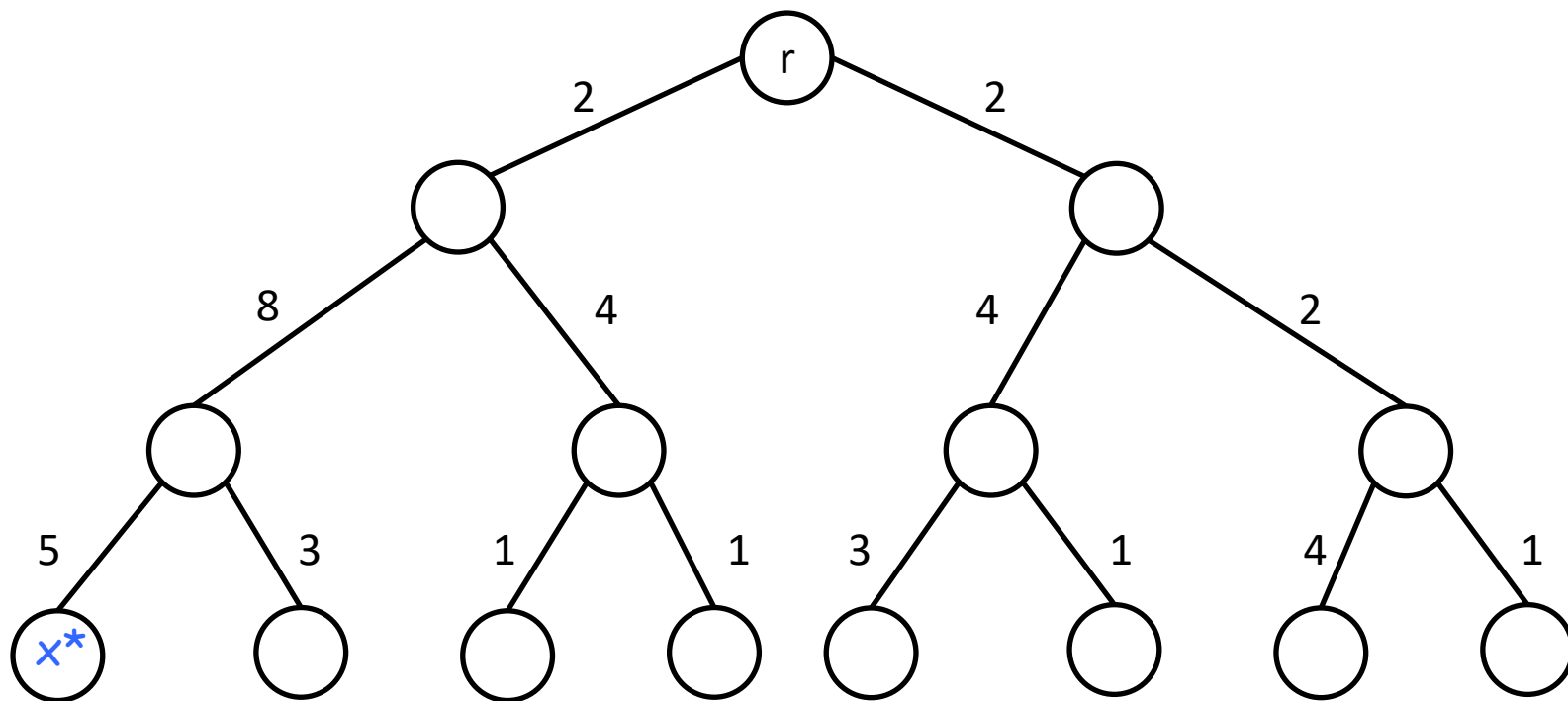
intuizione: conviene aumentare gli archi "alti"

definizione: arco e copre una foglia x se il cammino dalla radice verso x passa per e

F : insieme delle foglie

F_e : insieme delle foglie coperte da e

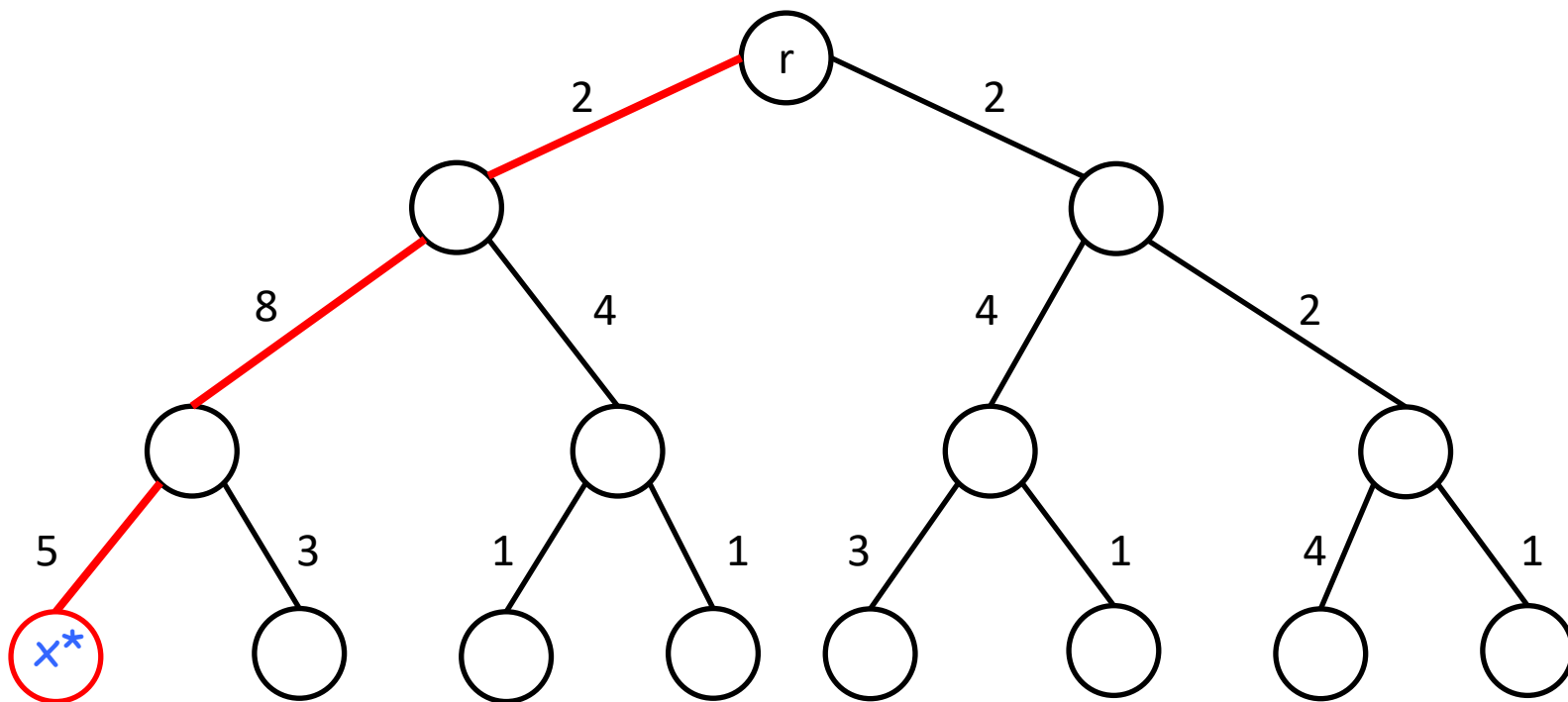
per ogni sottoinsieme di archi X
 $\{F_e\}_{e \in X}$ famiglia **laminare** di insiemi:
 per ogni e, e' in X
 - $F_e \subseteq F_{e'}$ oppure $F_e \cap F_{e'} = \emptyset$



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

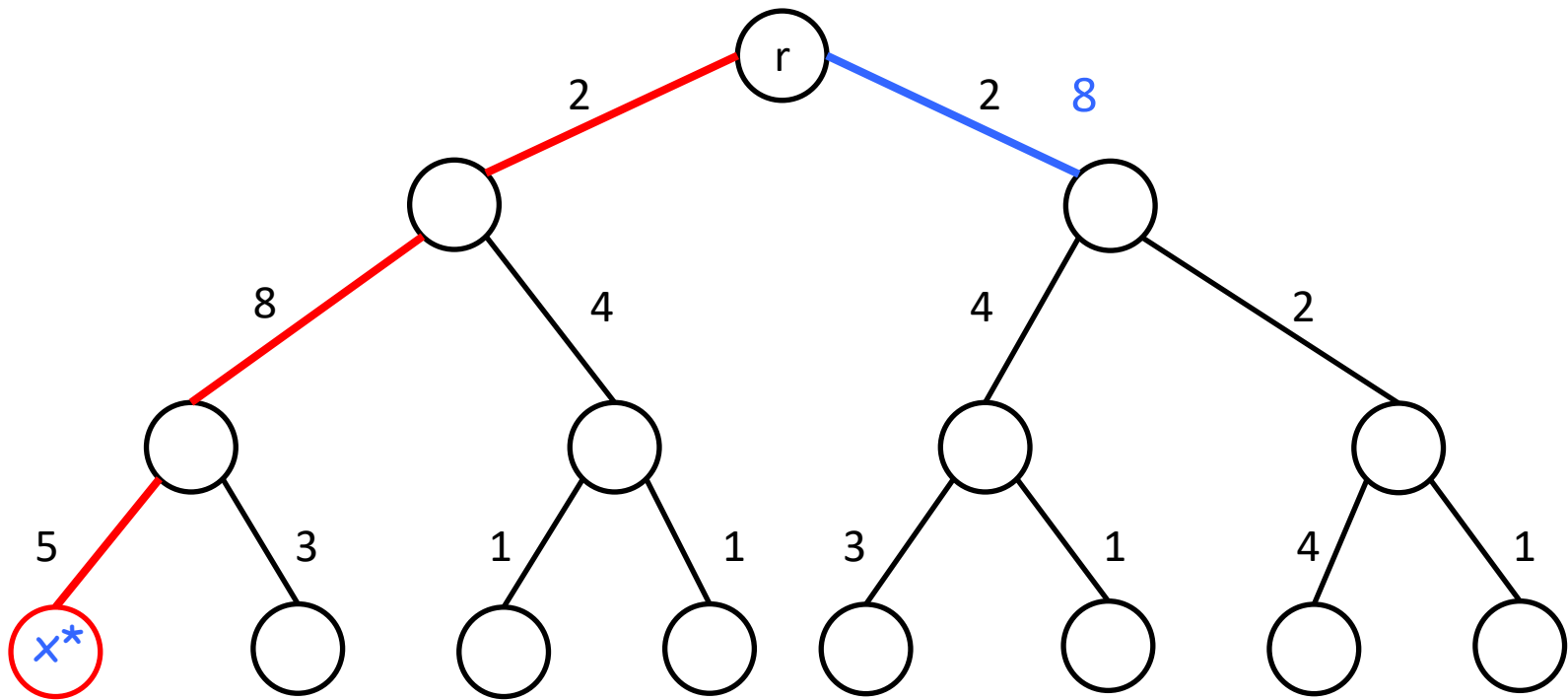
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

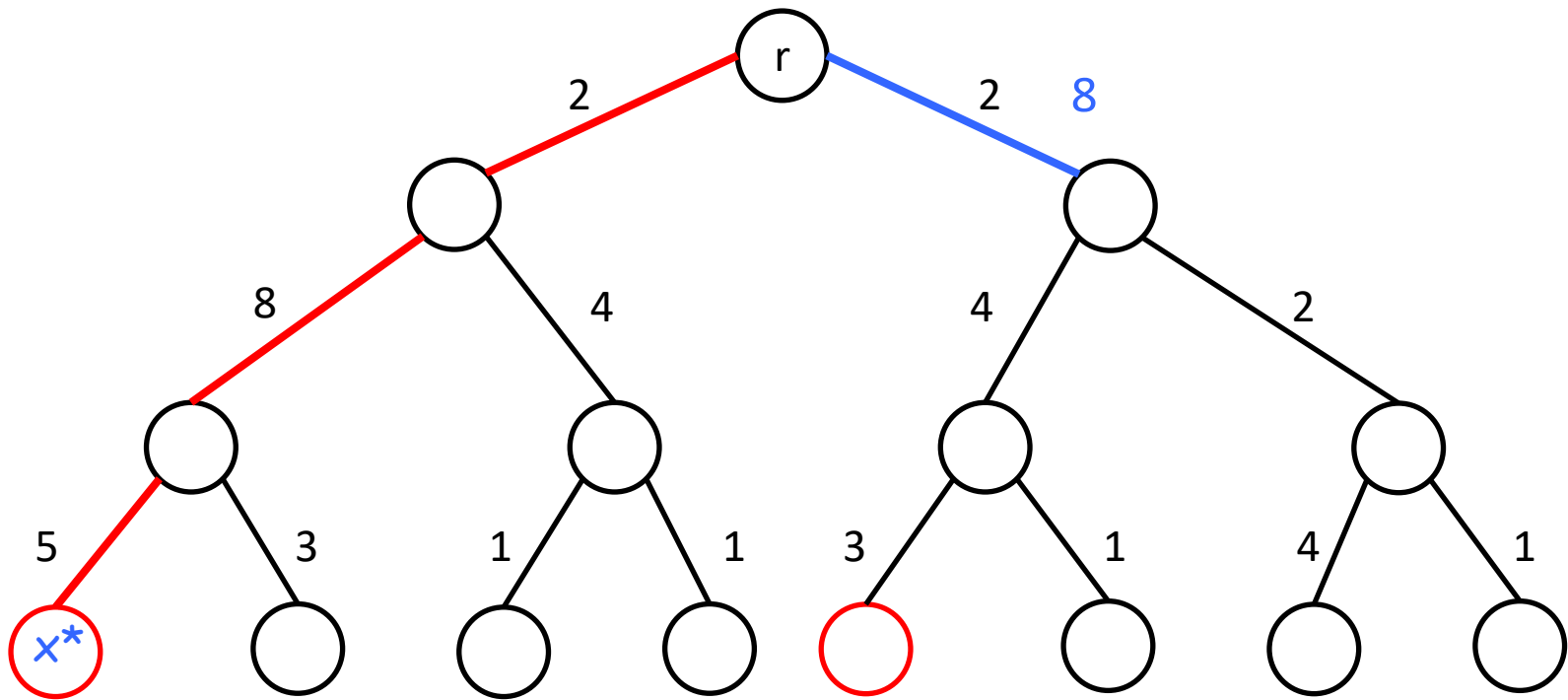
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

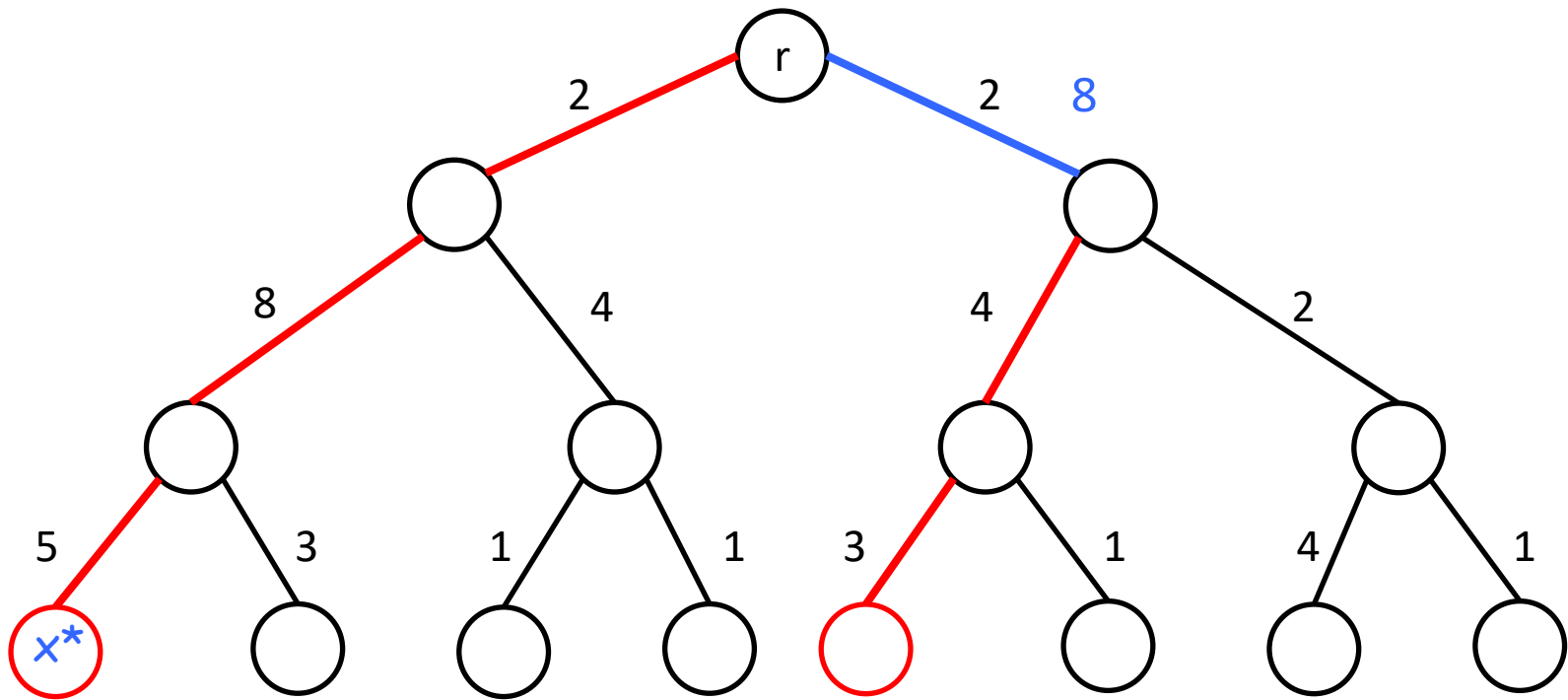
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

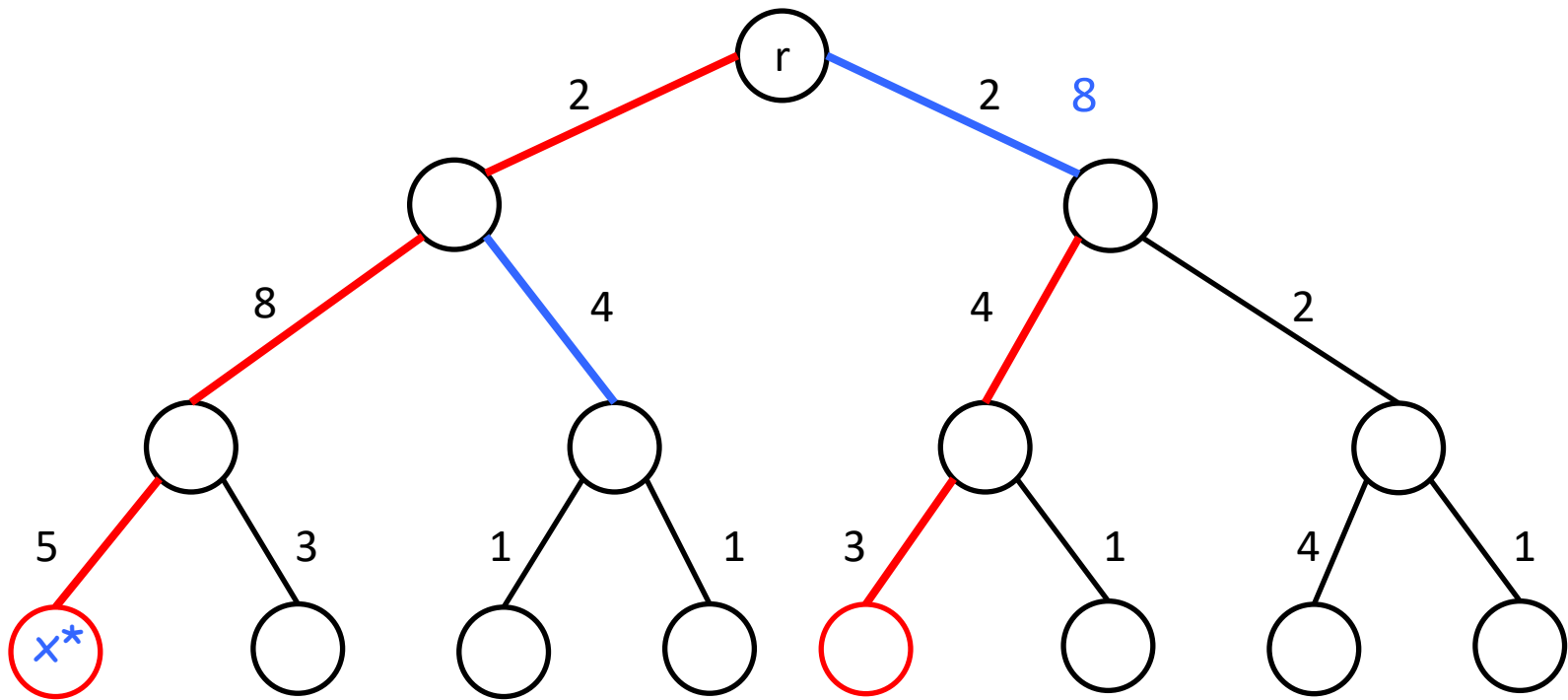
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

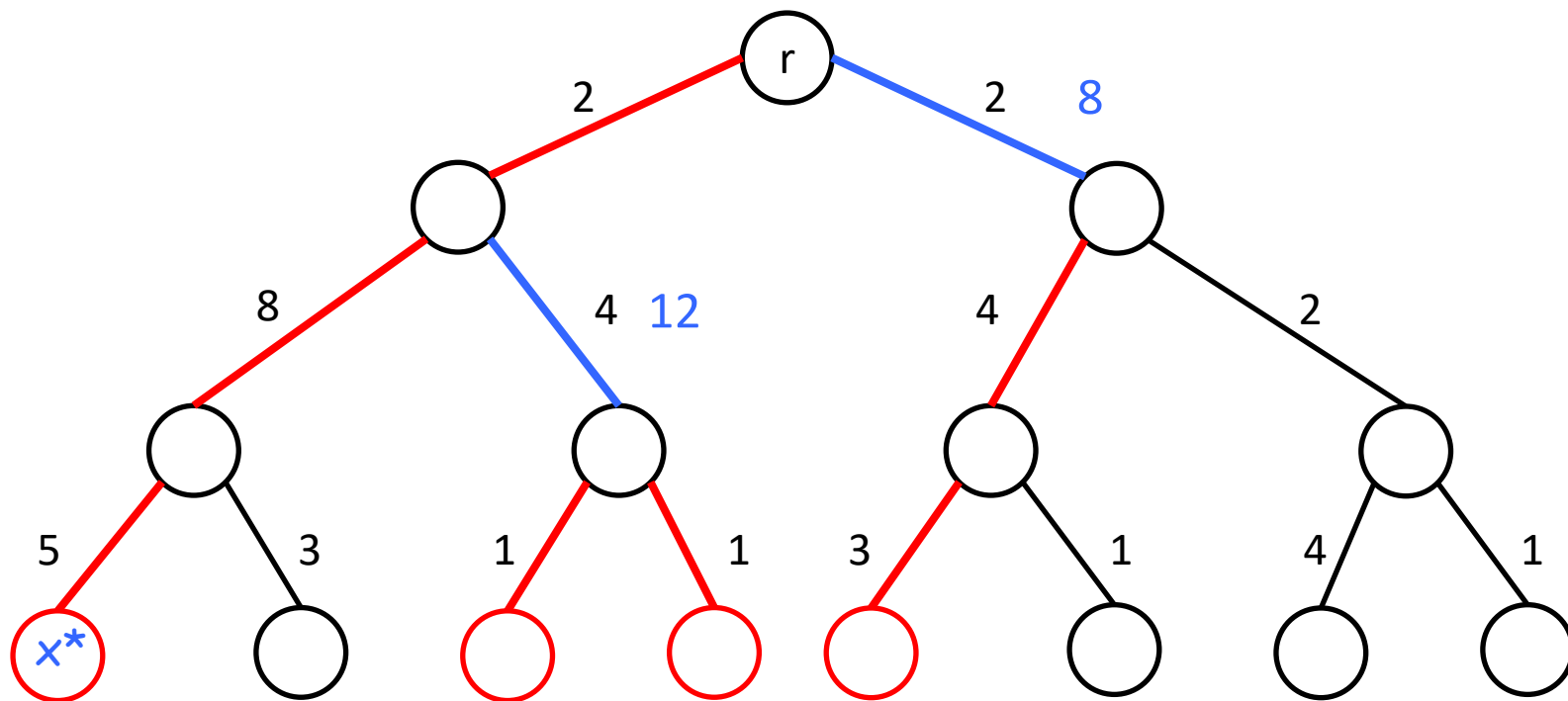
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

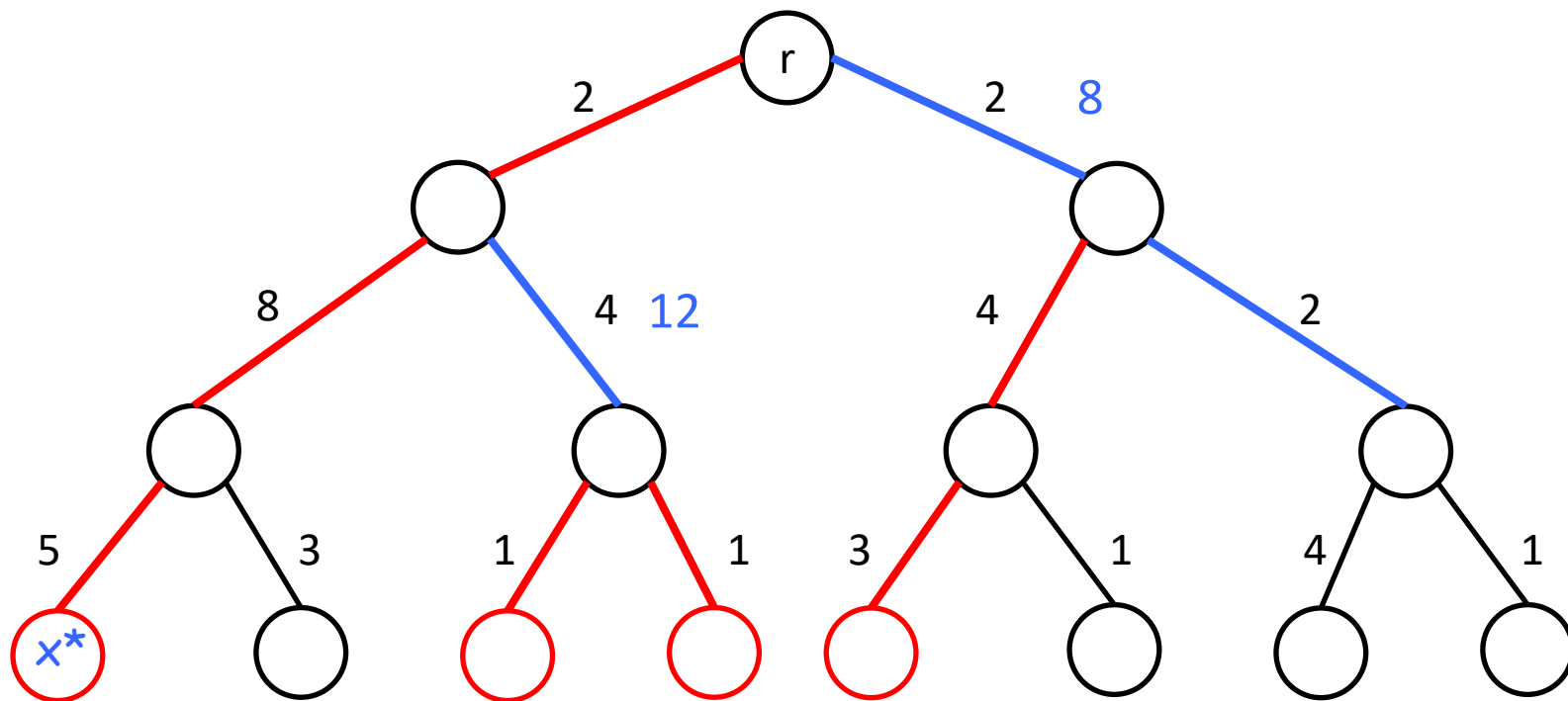
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

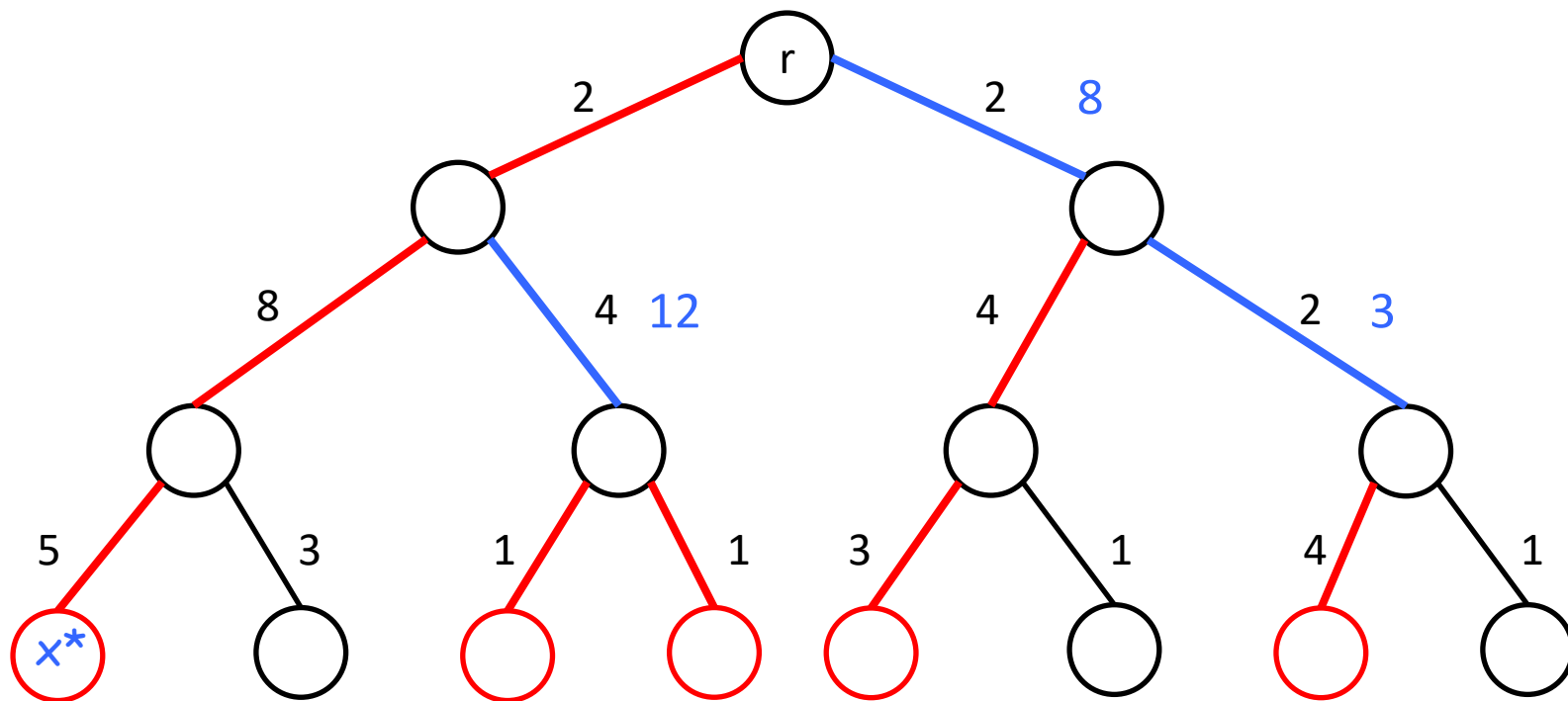
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

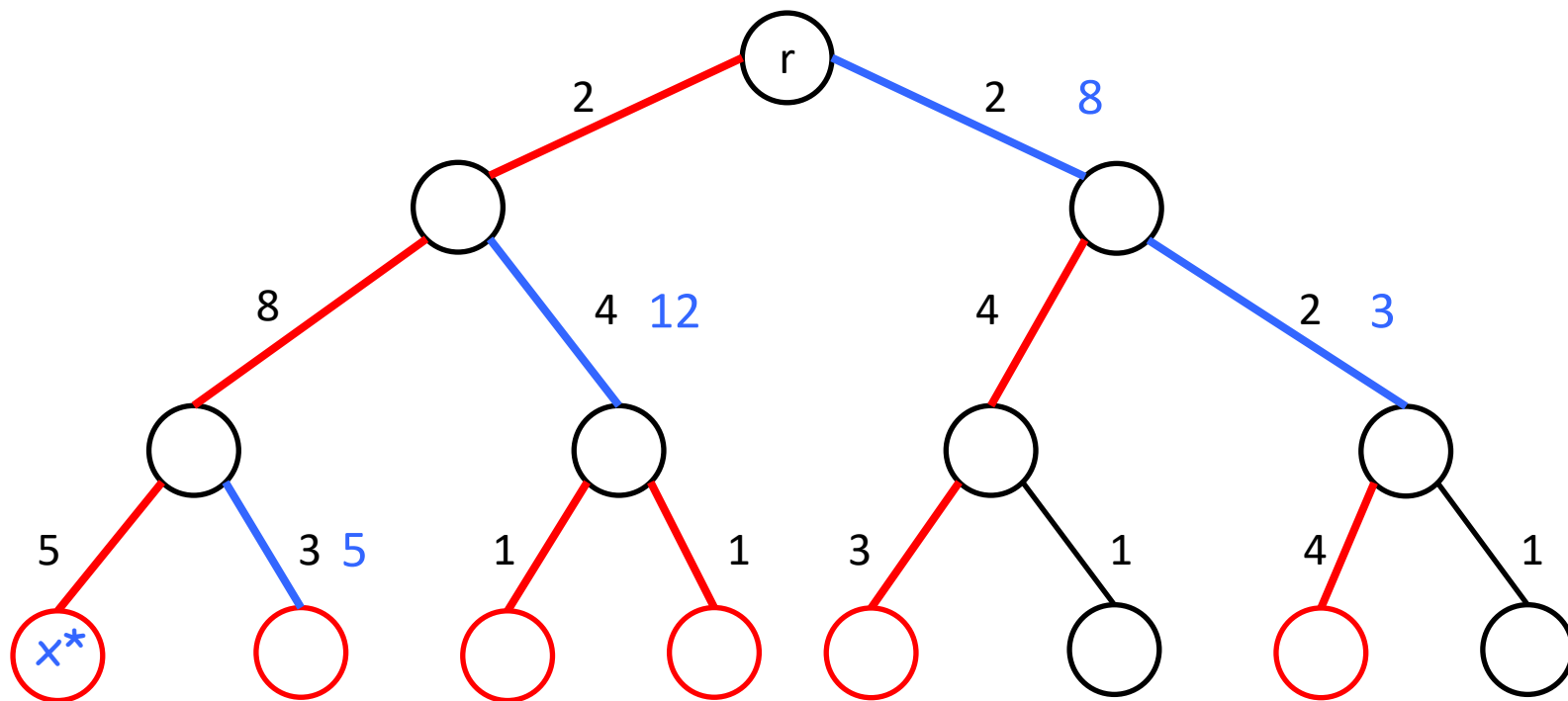
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

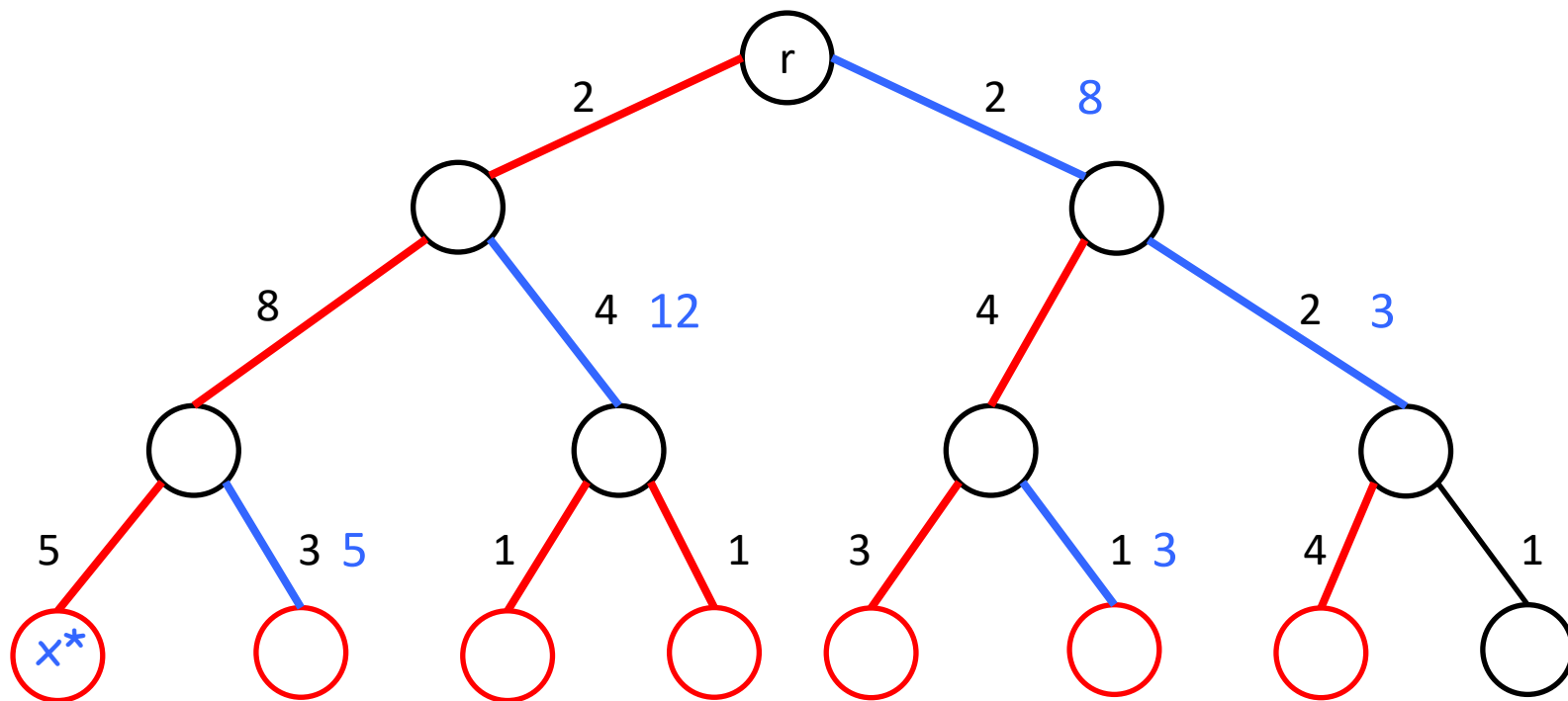
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

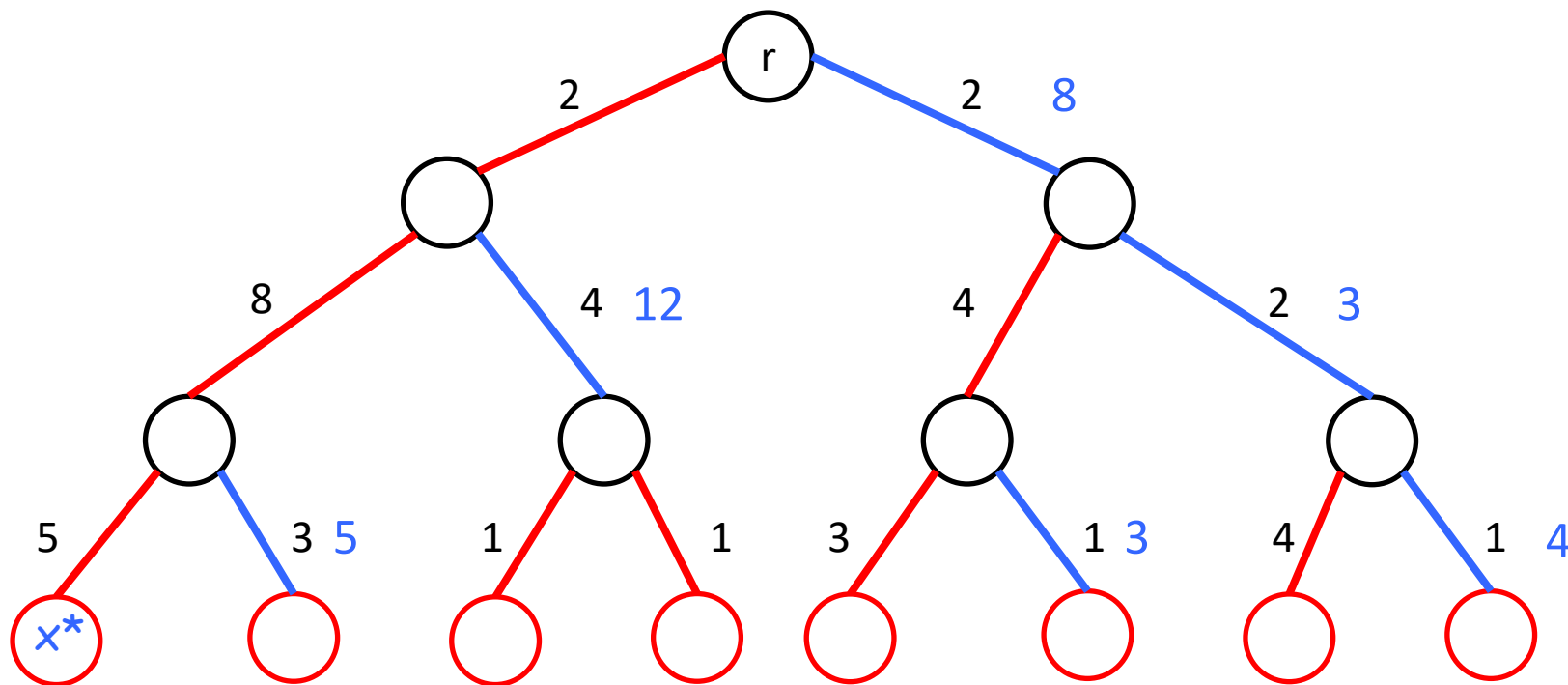
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

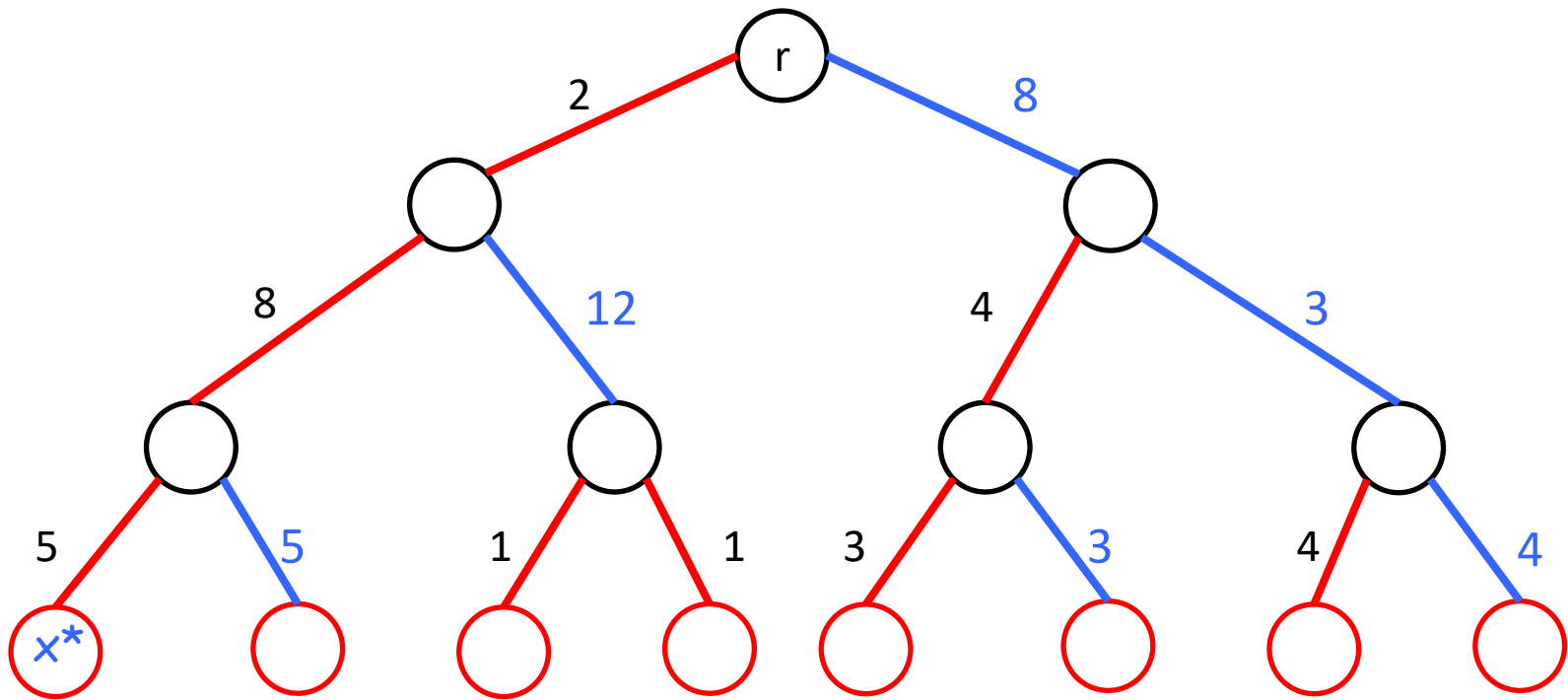
- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

algoritmo greedy:

- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura



x^* : foglia più lontana a distanza $L=15$

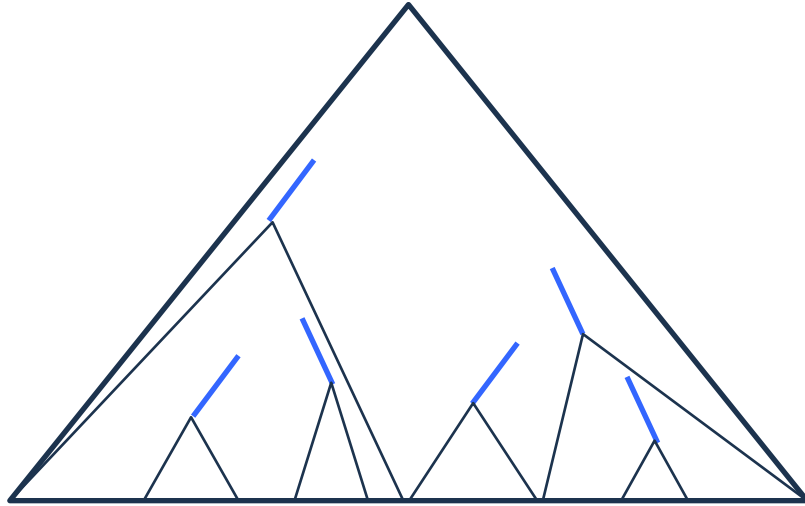
algoritmo greedy:

- marca tutti gli archi del cammino $r-x^*$
- considera gli archi di T top-down (in ordine ascendente di profondità)
 - se l'arco e non è marcato
 - alza il peso di e finché una foglia $x \in F_e$ non diventa a distanza L
 - marca tutti gli archi lungo il cammino verso x
- restituisci la nuova pesatura

dimostrazione di ottimalità

claim: una soluzione ottima mette tutte le foglie a distanza L

supponi Opt mette tutte le foglie a distanza $L' > L$



X : archi incrementati da Opt

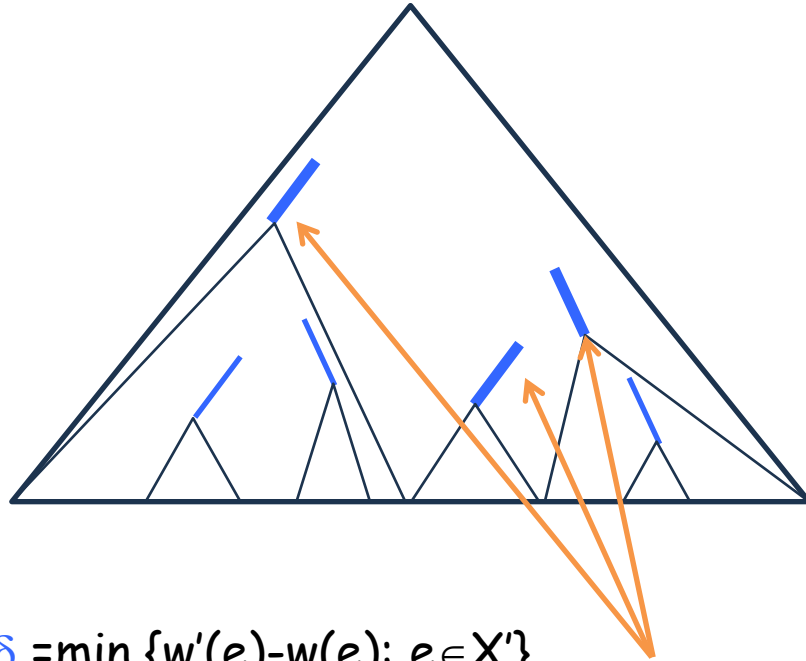
poiché tutte le foglie hanno
aumentato la loro distanza:

$$\bigcup_{e \in X} F_e = F$$

$X' \subseteq X$: tale che ogni foglia è
coperta da un solo $e \in X'$

claim: una soluzione ottima mette tutte le foglie a distanza L

supponi Opt mette tutte le foglie a distanza $L' > L$



X : archi incrementati da Opt

poiché tutte le foglie hanno aumentato la loro distanza:

$$\bigcup_{e \in X} F_e = F$$

$X' \subseteq X$: tale che ogni foglia è coperta da un solo $e \in X'$

$$\delta = \min \{w'(e) - w(e) : e \in X'\}$$

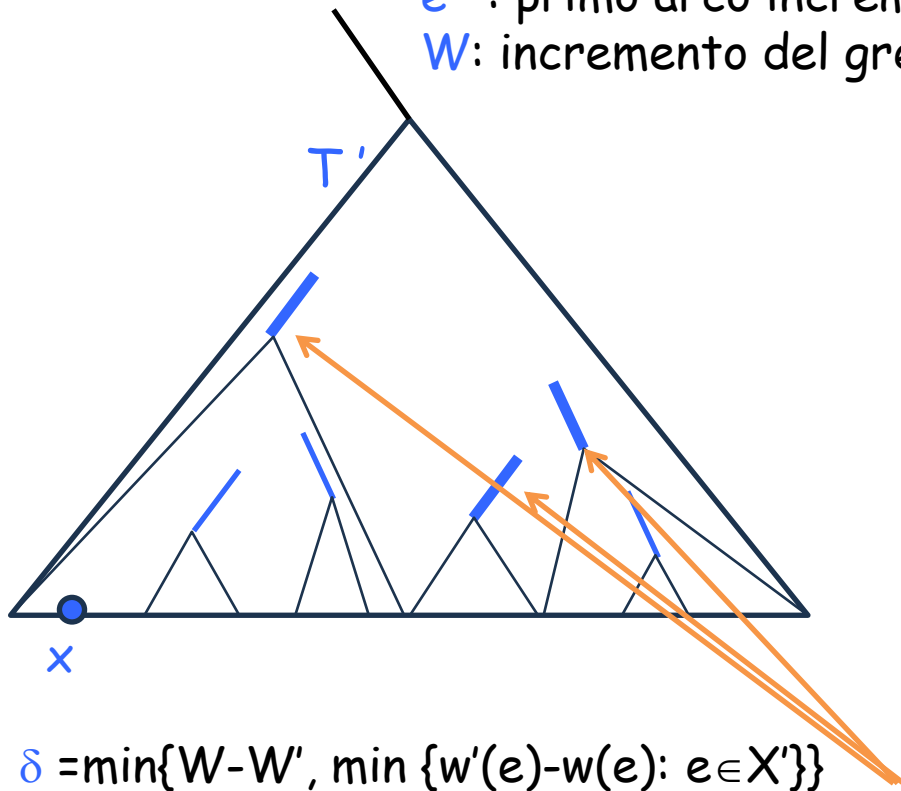
decremento di δ e ottengo una soluzione ammissibile strettamente migliore

allora Opt non era ottima: assurdo!

idea: faccio vedere che il greedy non sbaglia mai

e^* : primo arco incrementato dal greedy

W : incremento del greedy che porta x a distanza L



assumi Opt incrementa e^* di $W' < W$

X : archi incrementati da Opt in T'

poiché tutte le foglie di T' devono aumentare la loro distanza:

$$\bigcup_{e \in X} F_e = \text{foglie di } T'$$

$X' \subseteq X$: tale che ogni foglia di T' è coperta da un solo $e \in X'$

nota: $|X'| \geq 2$

$$\delta = \min\{W-W', \min\{w'(e)-w(e): e \in X'\}\}$$

allora Opt non era
ottima: assurdo!

decremento di δ e incremento e^* di δ
ottengo una soluzione ammissibile strettamente
migliore

Quindi l'ottimo deve incrementare e^* come il greedy: stesse argomentazioni per i prossimi archi.