Dispense per il Laboratorio di Strutture Dati e Algoritmi

Federico Bolelli

Esercitazione 04: Backtracking - Parte 2

Ultimo aggiornamento: 24/03/2022

Esercizio «Ombrelloni»

Visualizzare lo Spazio delle Soluzioni

Per risolvere un problema attraverso un algoritmo di *backtracking* è sempre opportuno:

1. Definire il dominio dei valori ammissibili per gli elementi della sequenza risolutiva;
2. Definire la lunghezza massima della sequenza che rappresenta una soluzione;
3. Definire, per ogni posizione della sequenza risolutiva, le eventuali regole matematiche che la soluzione parziale deve soddisfare. In altre parole, occorre definire i vincoli che sussistono tra gli elementi della sequenza risolutiva;
4. Rappresentare lo spazio delle soluzioni che la funzione deve esplorare per individuare i vettori soluzione.

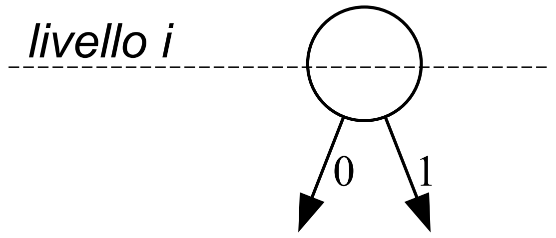
Visualizzare lo Spazio delle Soluzioni

Per risolvere un problema attraverso un algoritmo di *backtracking* è sempre opportuno:

1. Definire il dominio dei valori ammissibili per gli elementi della sequenza risolutiva;
2. Definire la lunghezza massima della sequenza che rappresenta una soluzione;
3. Definire, per ogni posizione della sequenza risolutiva, le eventuali regole matematiche che la soluzione parziale deve soddisfare. In altre parole, occorre definire i vincoli che sussistono tra gli elementi della sequenza risolutiva;
4. Rappresentare lo spazio delle soluzioni che la funzione deve esplorare per individuare i vettori soluzione.

Dominio dei Valori Ammissibili

- *Quante e quali scelte posso fare ad ogni passo?*
- Come per l'esercizio *SubsetK* il problema può essere modellato utilizzando un albero binario. Ad ogni passo devo decidere se posizionare o meno un ragazzo nell'ombrellone corrente.
- Ogni nodo dell'albero avrà quindi due rami uscenti:
  - (0) lascio libero l'ombrellone;
  - (1) ci posiziono un ragazzo.



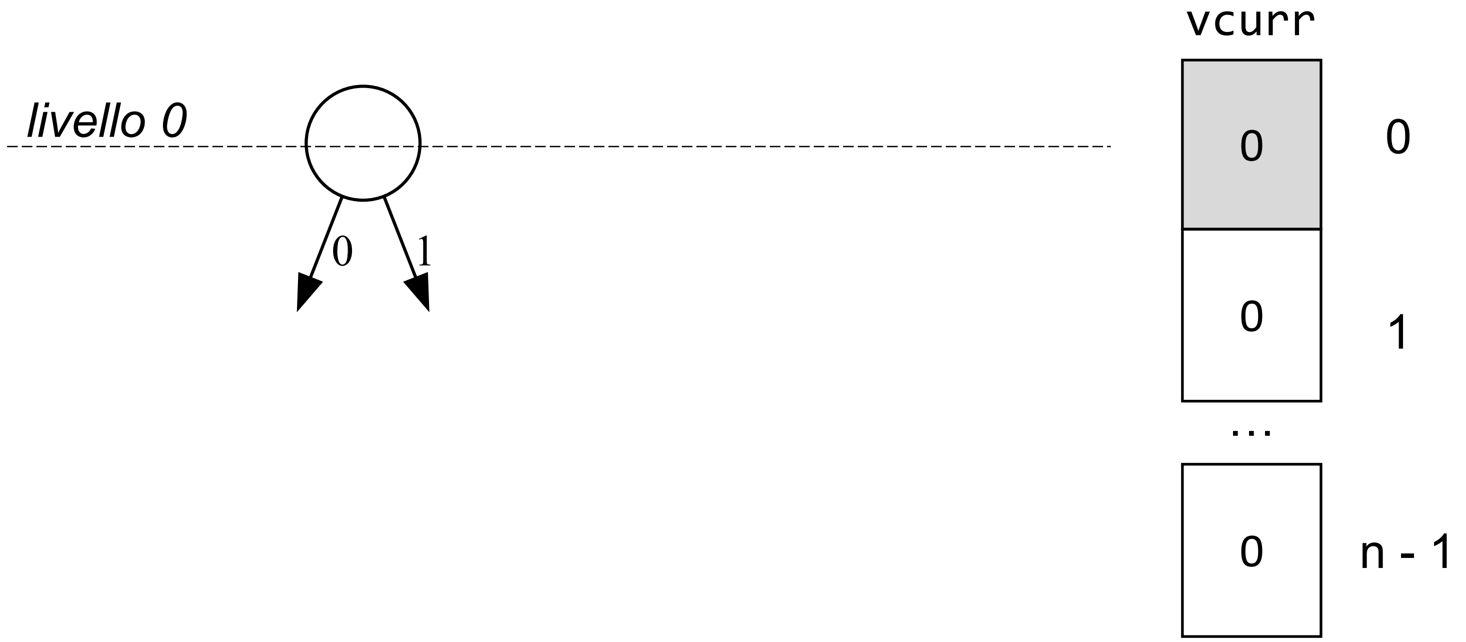
Facciamo il Punto

Per risolvere un problema attraverso un algoritmo di *backtracking* è sempre opportuno:

1. Definire il dominio dei valori ammissibili per gli elementi della sequenza risolutiva;
2. Definire la lunghezza massima della sequenza che rappresenta una soluzione;
3. Definire, per ogni posizione della sequenza risolutiva, le eventuali regole matematiche che la soluzione parziale deve soddisfare. In altre parole, occorre definire i vincoli che sussistono tra gli elementi della sequenza risolutiva;
4. Rappresentare lo spazio delle soluzioni che la funzione deve esplorare per individuare i vettori soluzione.

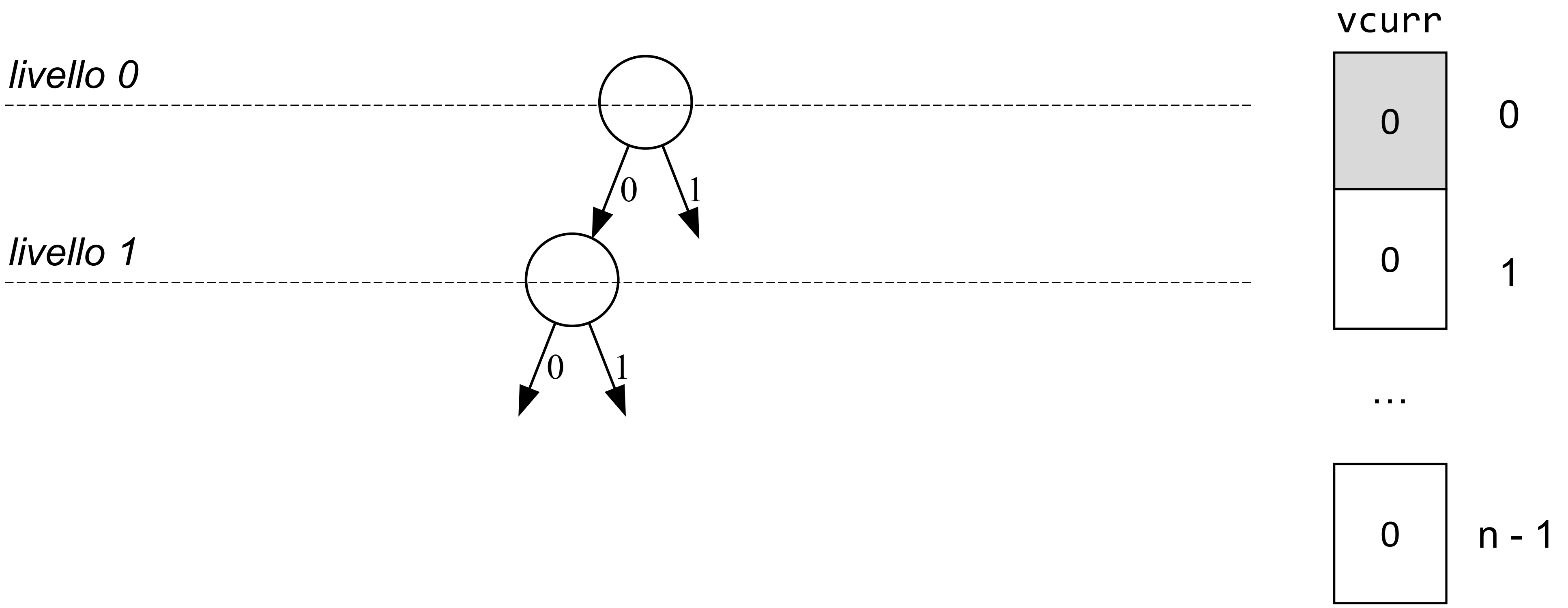
Lunghezza della Soluzione

- Il vettore «soluzione corrente», `vcurr`, deve poter contenere tante scelte quante sono gli ombrelloni in prima fila, ovvero `n`.
- All'inizio gli ombrelloni dovranno essere tutti liberi.
- Il livello corrente dell'albero, `i`, corrisponde alla posizione *i-esima* di `vcurr`



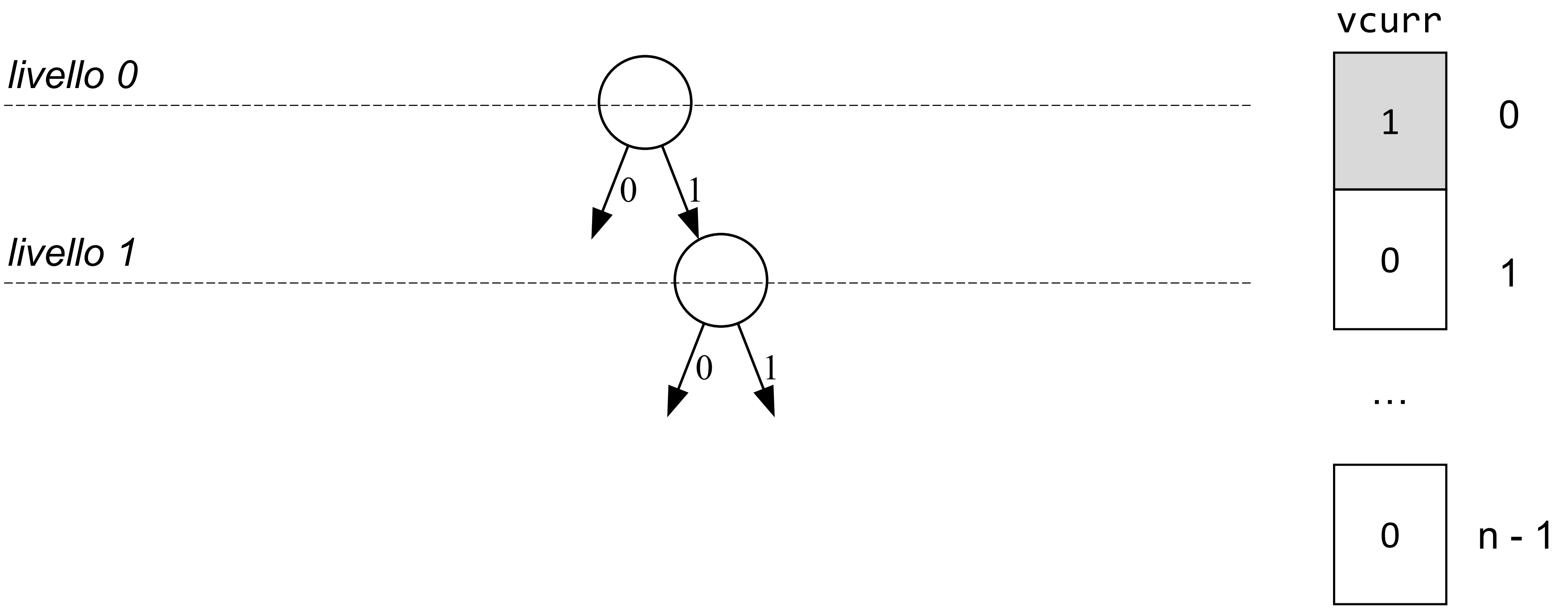
Lunghezza della Soluzione

- «Scendere» ricorsivamente sul ramo 0 significa lasciare l'ombrellone corrente vuoto e passare al successivo:



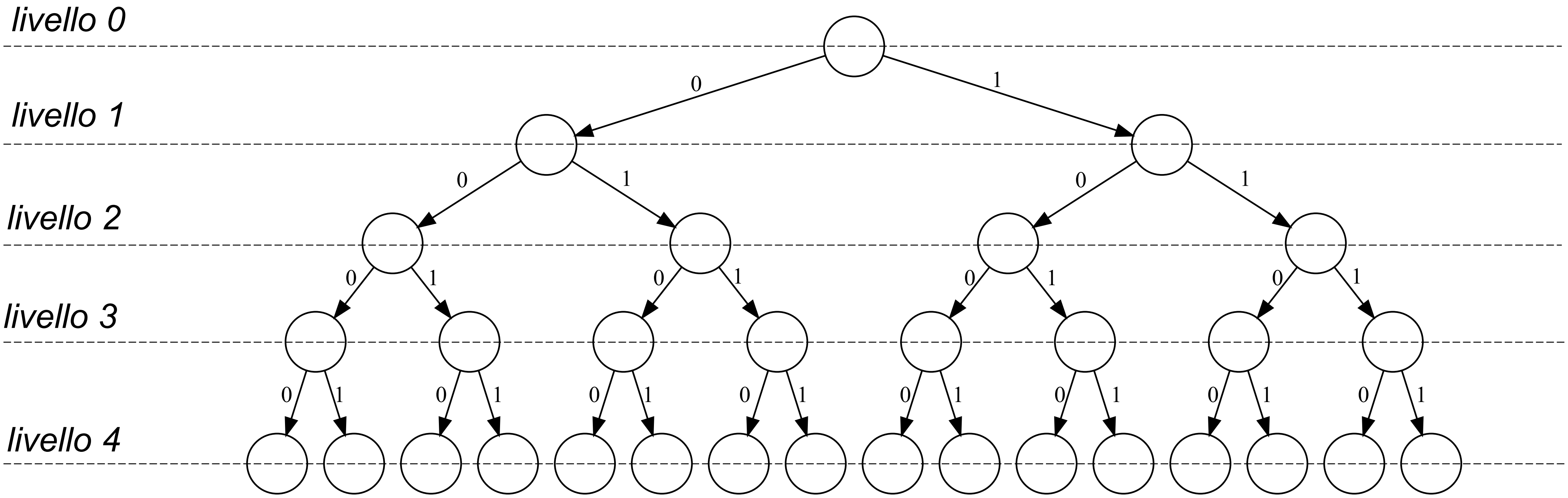
Lunghezza della Soluzione

- «Scendere» ricorsivamente sul ramo 1 significa posizionare un ragazzo nell'ombrellone corrente e passare al successivo:



Lunghezza della Soluzione

- *Quanti sono i livelli dell'albero?*
- n + 1 (da 0 a n)
- **Prendiamo ad esempio il caso n = 4:**



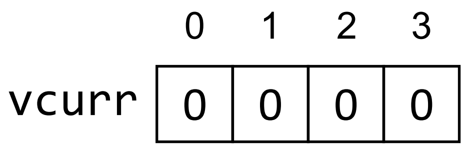
Visualizzare lo Spazio delle Soluzioni

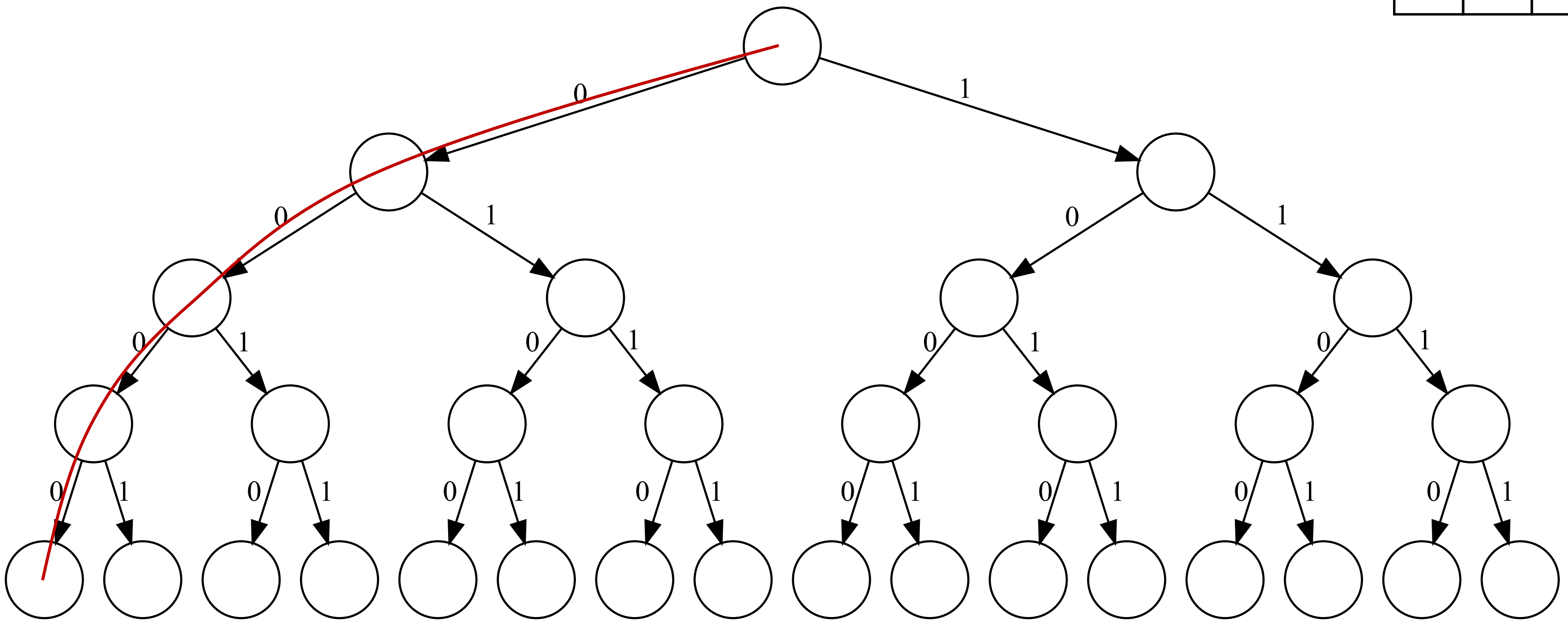
Per risolvere un problema attraverso un algoritmo di *backtracking* è sempre opportuno:

1. Definire il dominio dei valori ammissibili per gli elementi della sequenza risolutiva;
2. Definire la lunghezza massima della sequenza che rappresenta una soluzione;
3. Definire, per ogni posizione della sequenza risolutiva, le eventuali regole matematiche che la soluzione parziale deve soddisfare. In altre parole, occorre definire i vincoli che sussistono tra gli elementi della sequenza risolutiva;
4. Rappresentare lo spazio delle soluzioni che la funzione deve esplorare per individuare i vettori soluzione.

Soluzioni Non Ammissibili (vincoli)

- **Prendiamo ad esempio il caso n = 4 e k = 2:**
- *Tutte le foglie rappresentano soluzioni valide?*
- No!



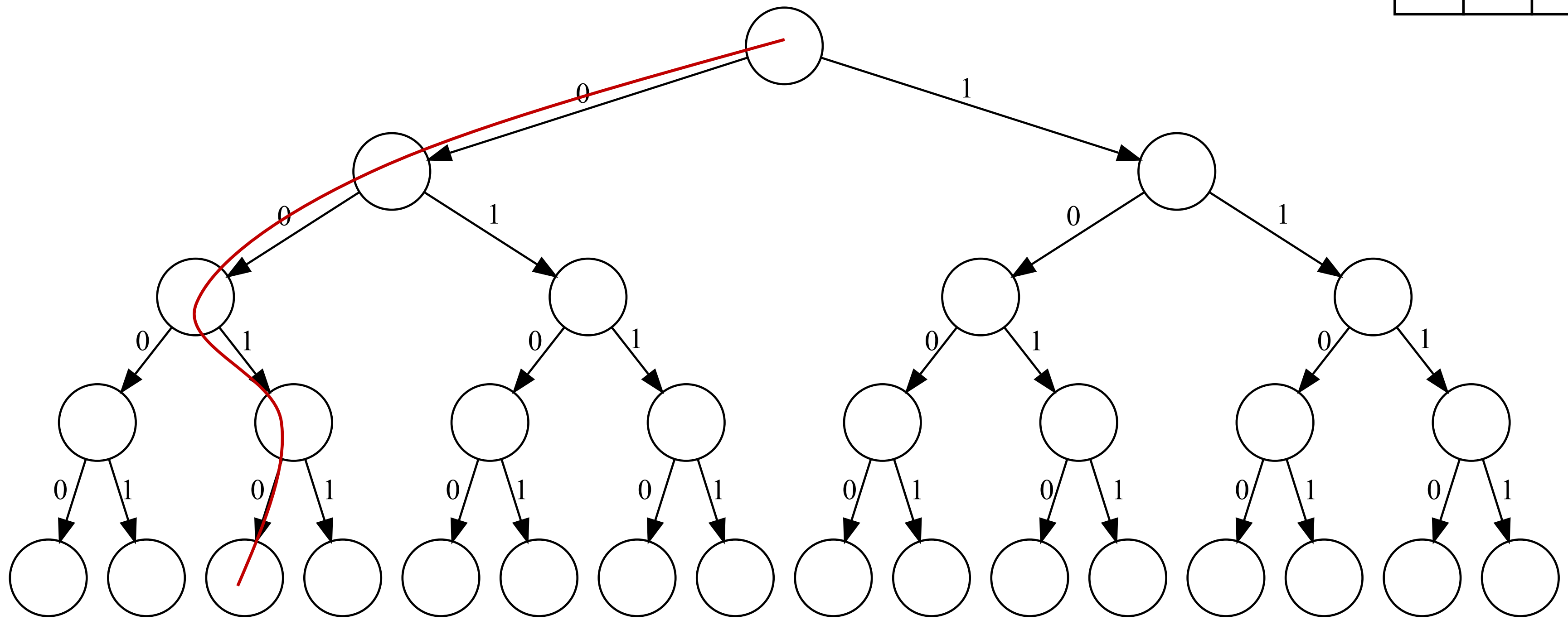


Questa «soluzione» non posiziona alcun ragazzo, quindi non è corretta!

Soluzioni Non Ammissibili (vincoli)

- **Prendiamo ad esempio il caso `n = 4 e k = 2`:**
- *Tutte le foglie rappresentano soluzioni valide?*
- No!

	0	1	2	3
vcurr	0	0	1	0

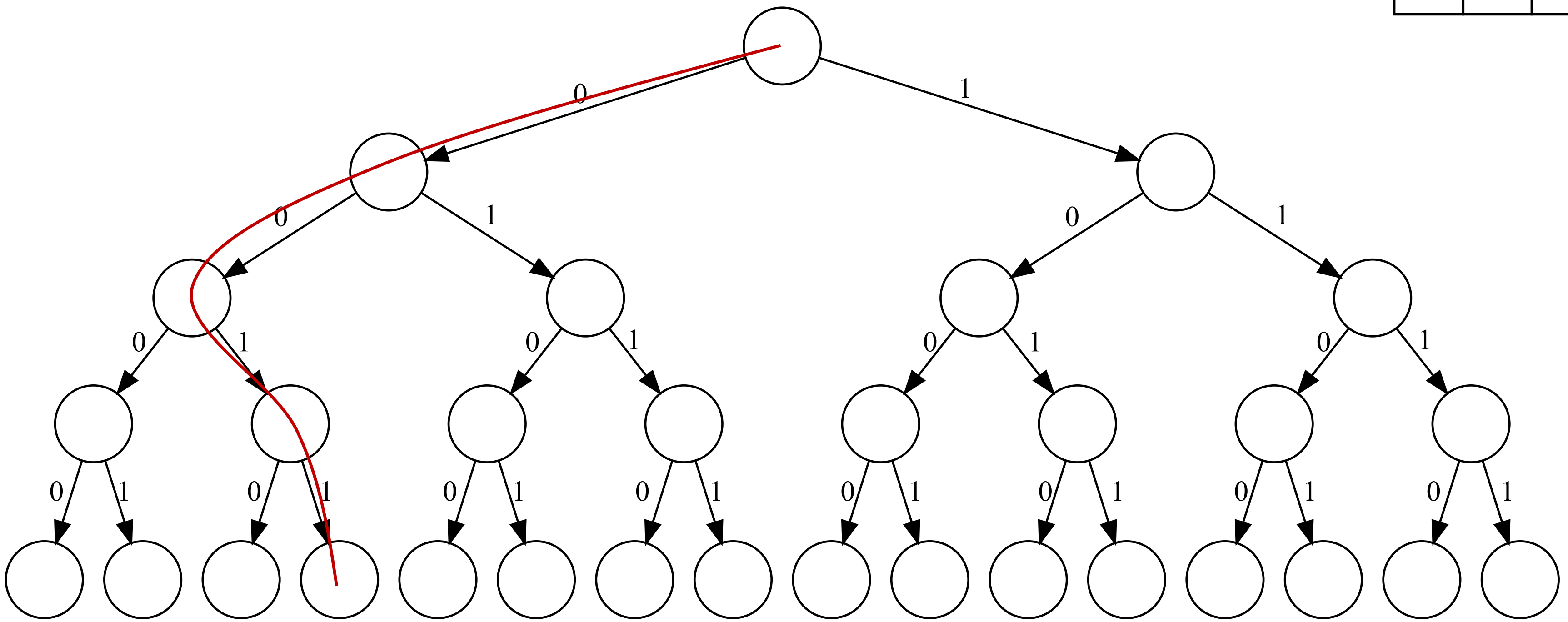


Questa «soluzione» posiziona un solo ragazzo, quindi non è corretta!

Soluzioni Non Ammissibili (vincoli)

- **Prendiamo ad esempio il caso `n = 4 e k = 2`:**
- *Tutte le foglie rappresentano soluzioni valide?*
- No!

	0	1	2	3
vcurr	0	0	1	1



Questa «soluzione» posiziona due ragazzi adiacenti, quindi non è corretta!

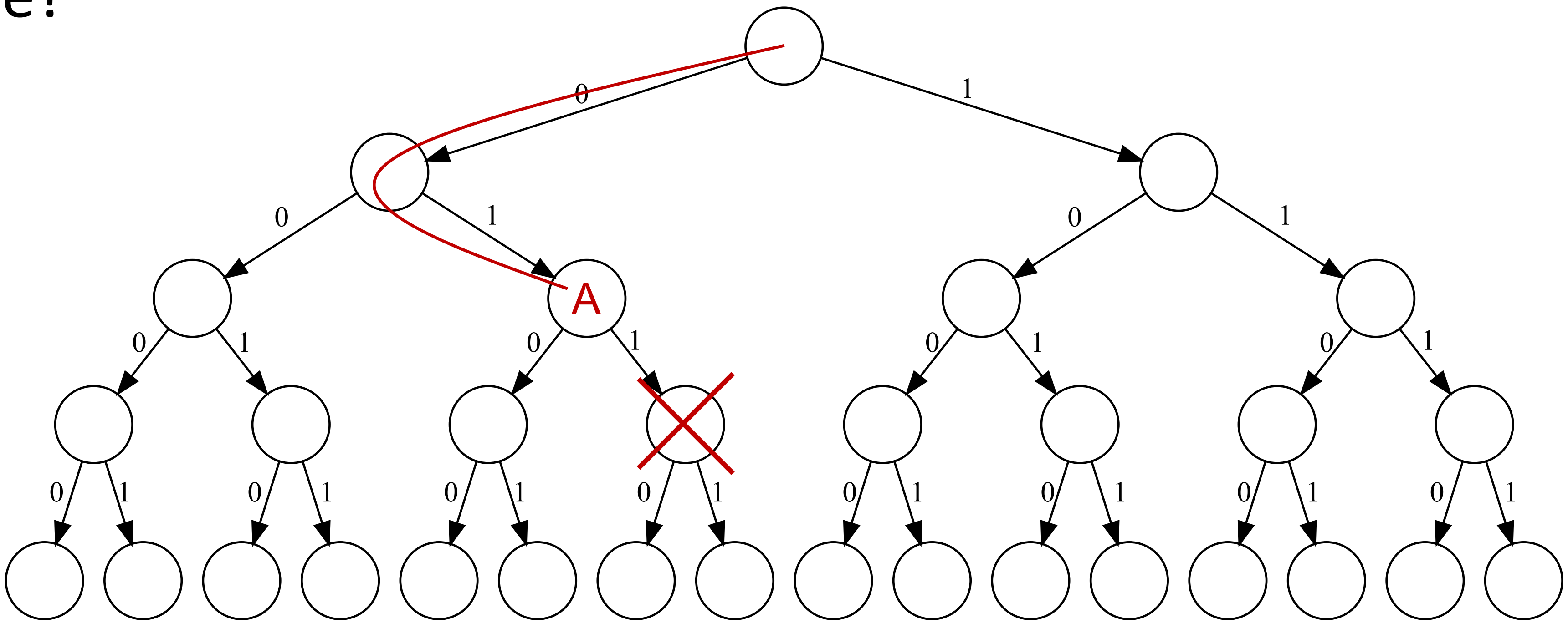
Passiamo all'Implementazione

- Quando arriviamo ad una foglia siamo di fronte ad un **caso base** e dobbiamo interrompere la ricorsione.
- Dobbiamo quindi verificare se la soluzione corrente è valida e, nel caso, stamparla su `stdout` come richiesto dall'esercizio.

Soluzione nel file `ombrelloni.c` su moodle.

Pruning

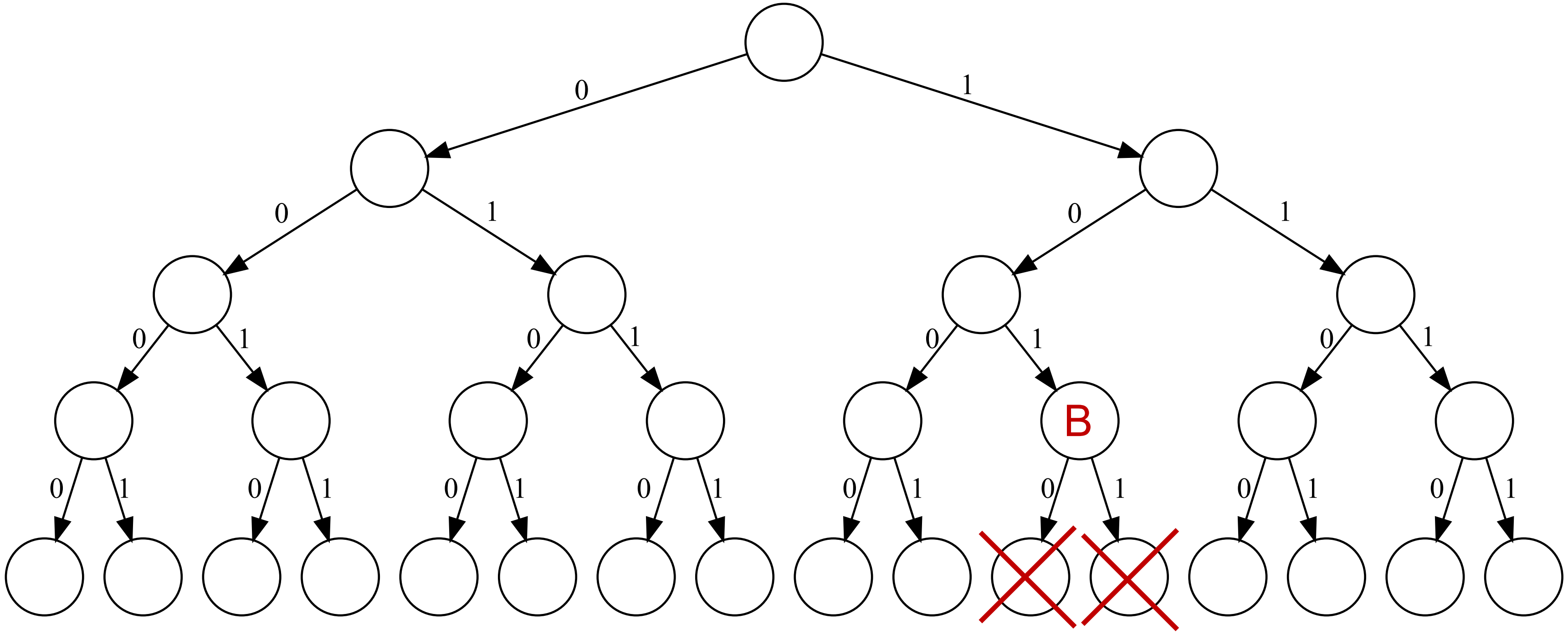
- **Prendiamo ad esempio il caso n = 4 e k = 2.**
- È sempre vero che ad ogni passo della ricorsione abbiamo due strade?
- No!



Ancor prima di esplorarlo, sappiamo che il ramo destro del nodo A porterà ad una soluzione non valida: quindi possiamo eliminare l'intero sottoalbero!

Pruning

- **Prendiamo ad esempio il caso n = 4 e k = 2.**
- È necessario arrivare ad una foglia per trovare una soluzione?
- No!



Arrivati al nodo B sappiamo già che questa è una soluzione valida. Non ha senso proseguire!

Pruning

- Siamo ad un **caso base** quando abbiamo posizionato esattamente k ragazzi e quando non ci sono più ombrelloni da utilizzare.
- Nel primo caso, avendo già fatto tutte le verifiche richieste, possiamo limitarci a stampare la soluzione.
- Nel secondo, invece, sappiamo che la soluzione trovata non è valida perché non ci sono più ombrelloni disponibili e meno di k ragazzi sono stati posizionati.
- **In entrambi i casi dobbiamo ricordarci di interrompere le chiamate ricorsive.**

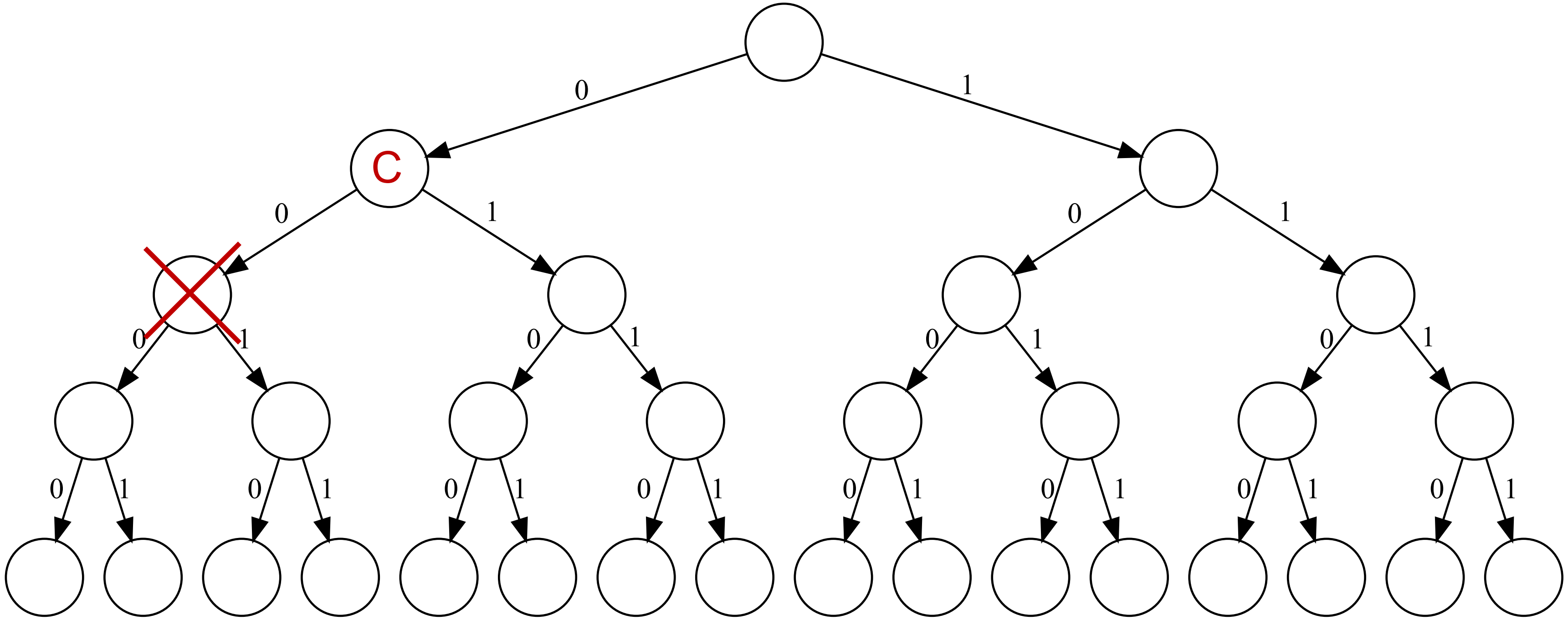
Soluzione nel file `ombrelloni_pruning.c` su moodle.

Pruning

- In realtà si può fare ancora meglio.

Pruning

- **Prendiamo ad esempio il caso n = 4 e k = 2.**
- È necessario arrivare ad una foglia per scartare una soluzione?
- No!



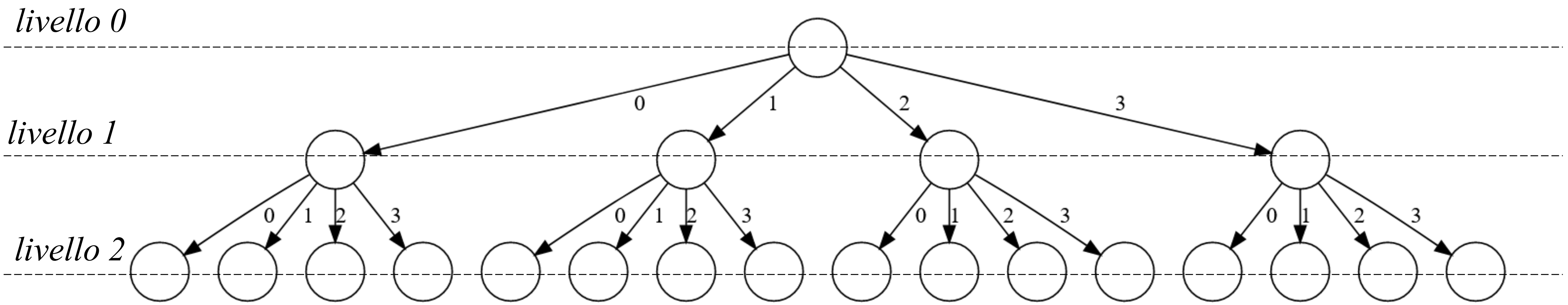
Arrivati al nodo C sappiamo che il ramo 0 porterà a una soluzione NON valida perché due ragazzi non possono essere posizionati in due ombrelloni adiacenti

Altri Modi per Modellare il Problema

- In questa soluzione abbiamo modellato lo spazio delle soluzioni utilizzando un albero binario. Ovvero ad ogni livello, i, decidiamo se posizionare un ragazzo nell'ombrellone i-esimo.
- Esistono altri modi per modellare questo spazio delle soluzioni?

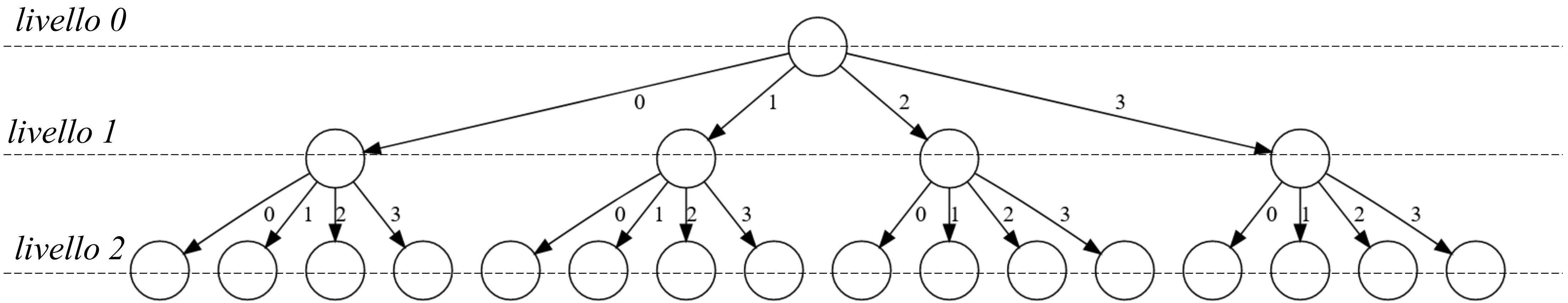
Altri Modi per Modellare il Problema

- Potremmo ad esempio utilizzare un albero *n*-ario, dove *n* è il numero di ombrelloni in prima fila.
- **Se prendiamo ad esempio il caso n = 4 e k = 2**, l'albero delle soluzioni può essere così raffigurato:



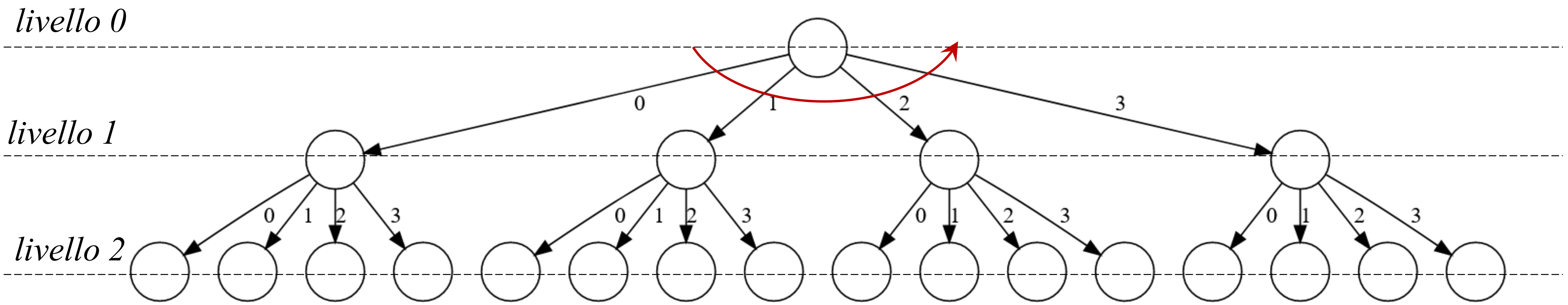
Dominio dei Valori Ammissibili

- Il livello dell'albero in cui mi trovo corrisponde al numero di ragazzi posizionati.
- Al primo passo della ricorsione decido in quale degli n ombrelloni posizionare il primo ragazzo, al secondo passo decido la posizione del secondo ragazzo e così via.



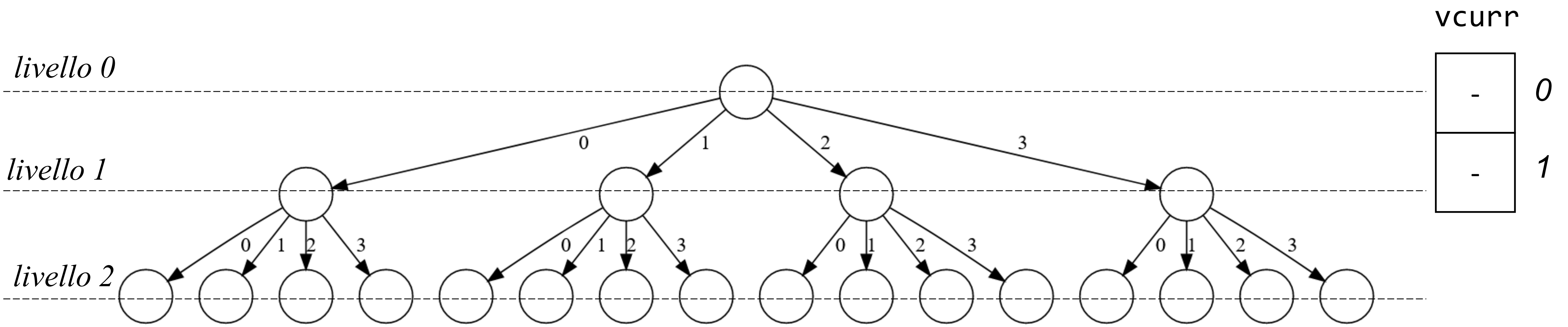
Dominio dei Valori Ammissibili

- Il dominio dei valori ammissibili è pari al numero di ombrelloni in prima fila: n
- Come già detto più volte, questo valore corrisponde al numero di figli che ogni nodo dell’albero delle soluzioni può avere.

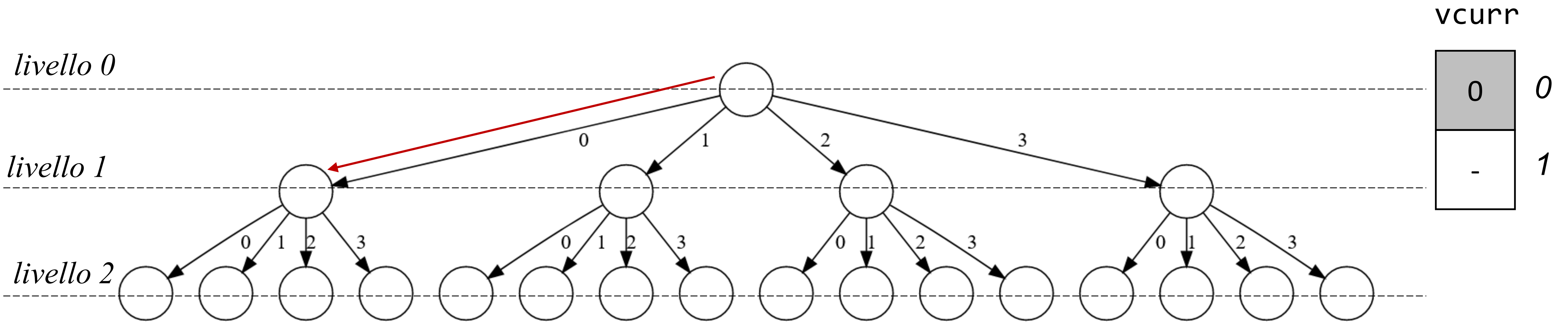


Lunghezza della Soluzione

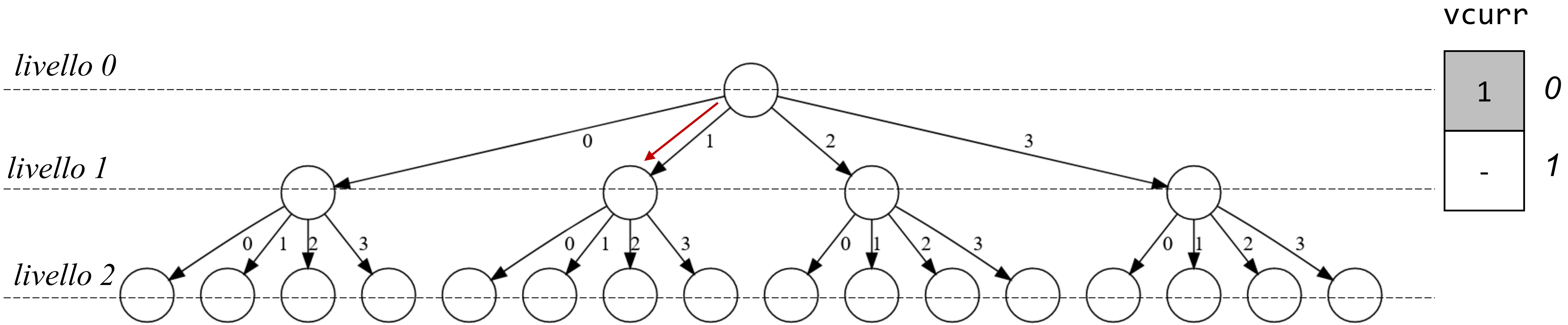
- Come al solito, memorizzo la scelta effettuata al passo i-esimo nella posizione i del vettore `vcurr`
- `vcurr` deve in questo caso contenere degli interi e avrà lunghezza pari al numero di ragazzi da posizionare: `k`



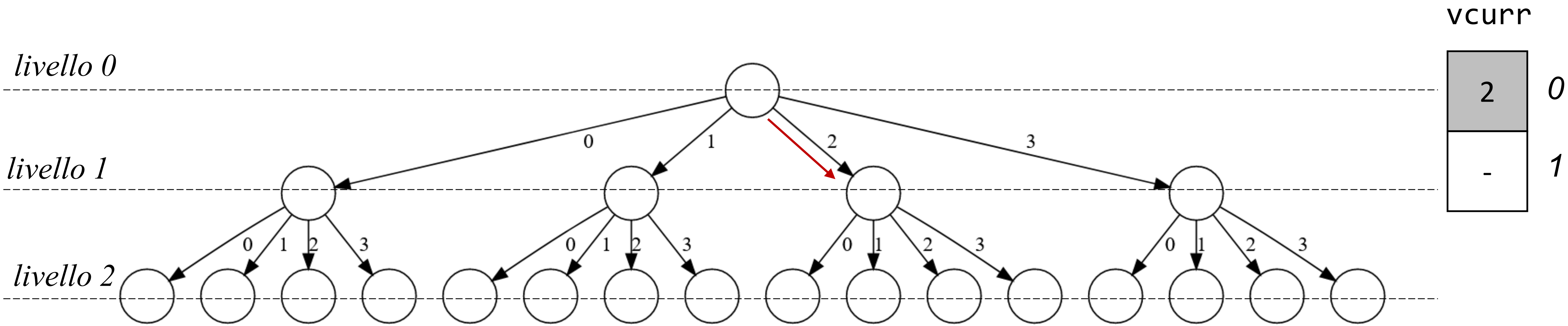
Visualizzazione



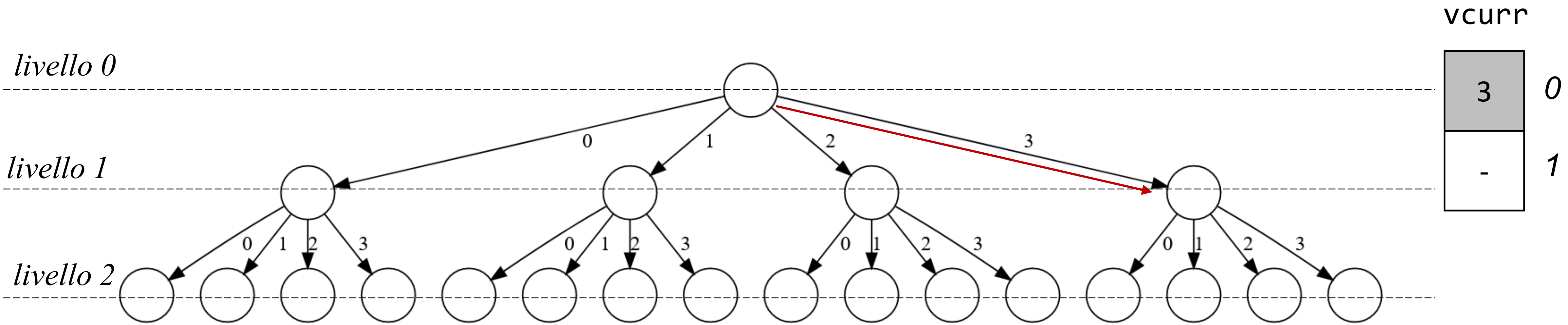
Visualizzazione



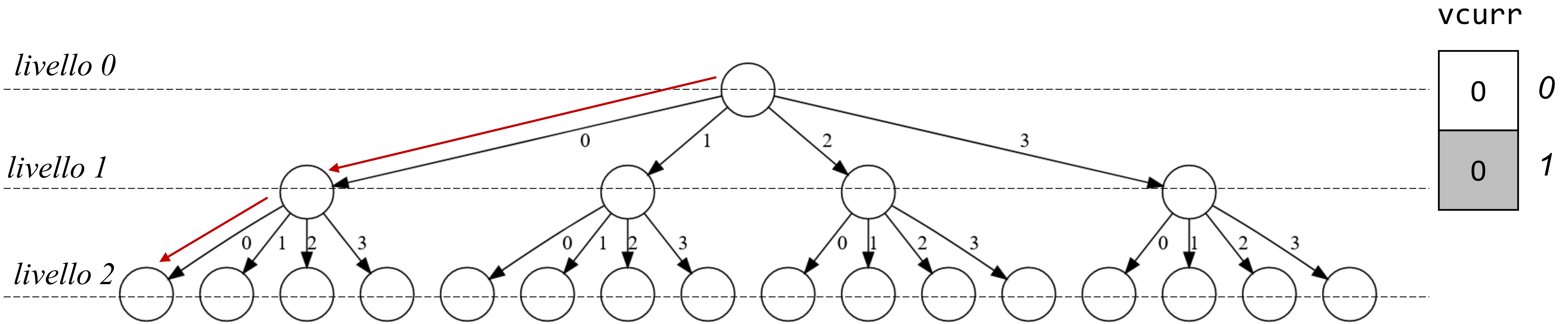
Visualizzazione



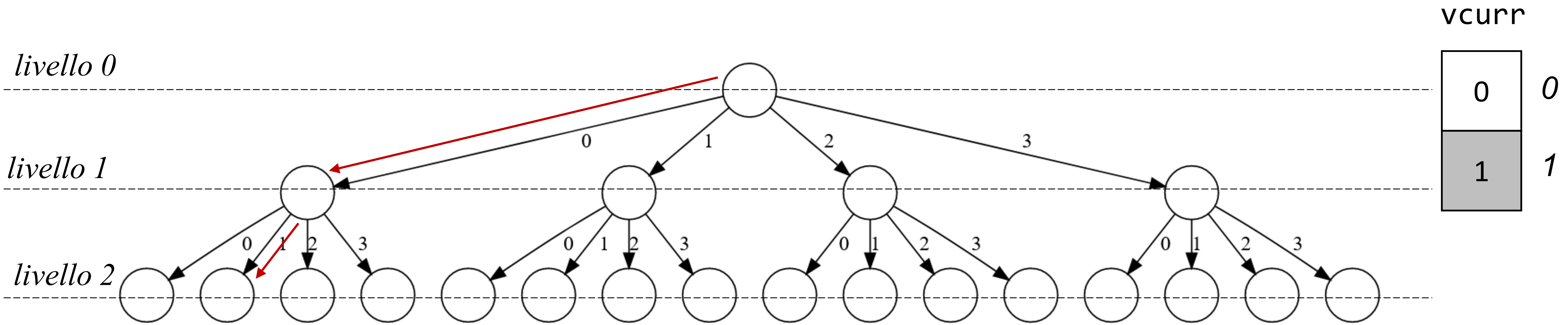
Visualizzazione



Visualizzazione

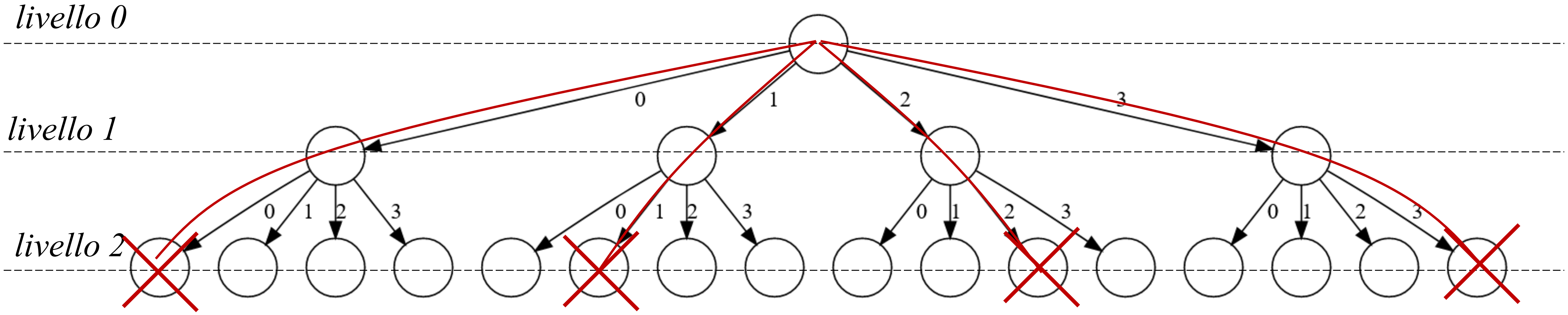


Visualizzazione



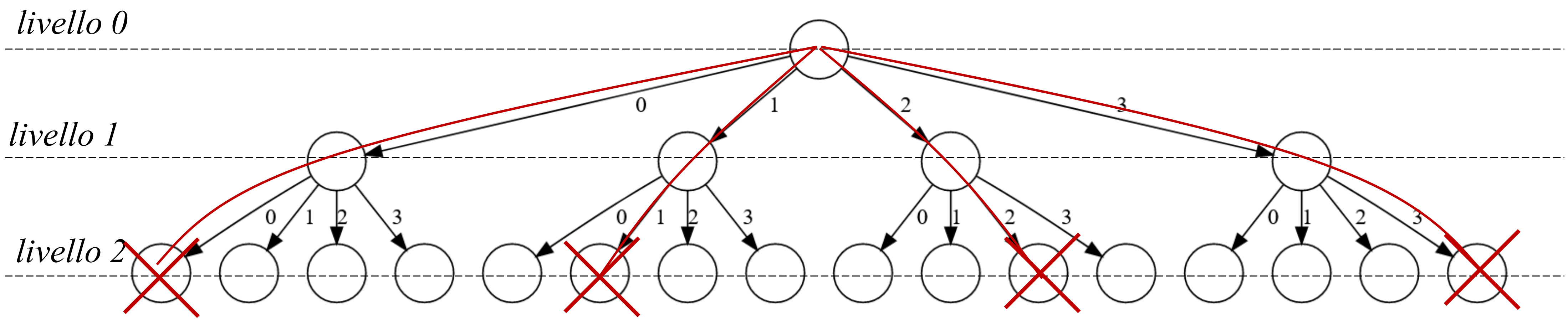
Soluzioni Non Ammissibili (vincoli)

- È evidente che l'esplorazione di questo albero porta a numerose soluzioni non valide.
- Non è ad esempio consentito posizionare più ragazzi sullo stesso ombrellone.



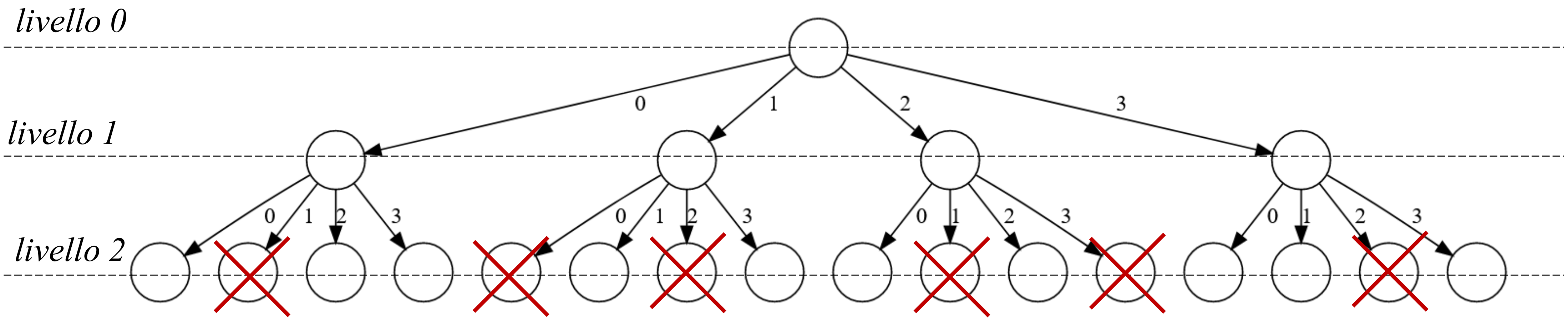
Soluzioni Non Ammissibili (vincoli)

- Nota che questo vincolo era intrinseco nella modellazione precedente e non doveva essere specificatamente gestito.
- In questo caso, se al passo i faccio la scelta x, questa dovrà essere scartata in tutti i passi successivi.



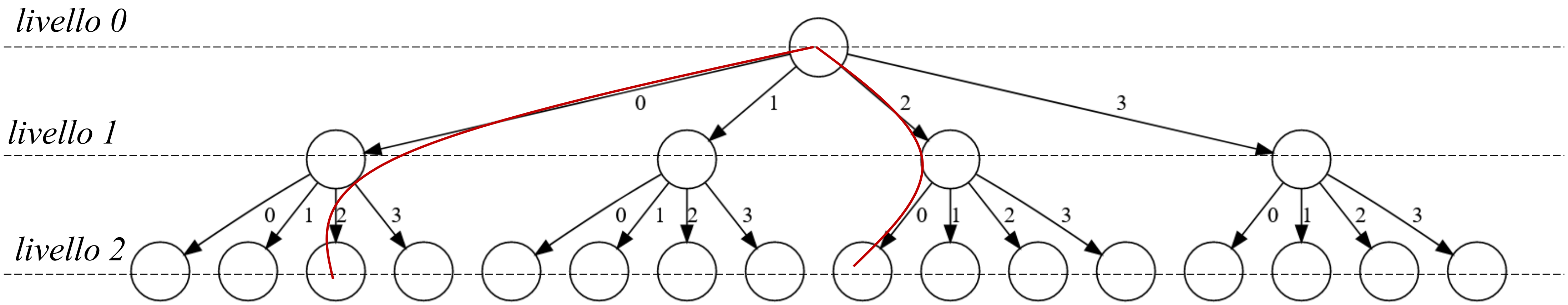
Soluzioni Non Ammissibili (vincoli)

- Le soluzioni che posizionano due ragazzi adiacenti sono da scartare.
- Quindi se al passo i ho fatto la scelta x, al passo $i + 1$ non potrò fare né la scelta $x - 1$, né la scelta $x + 1$



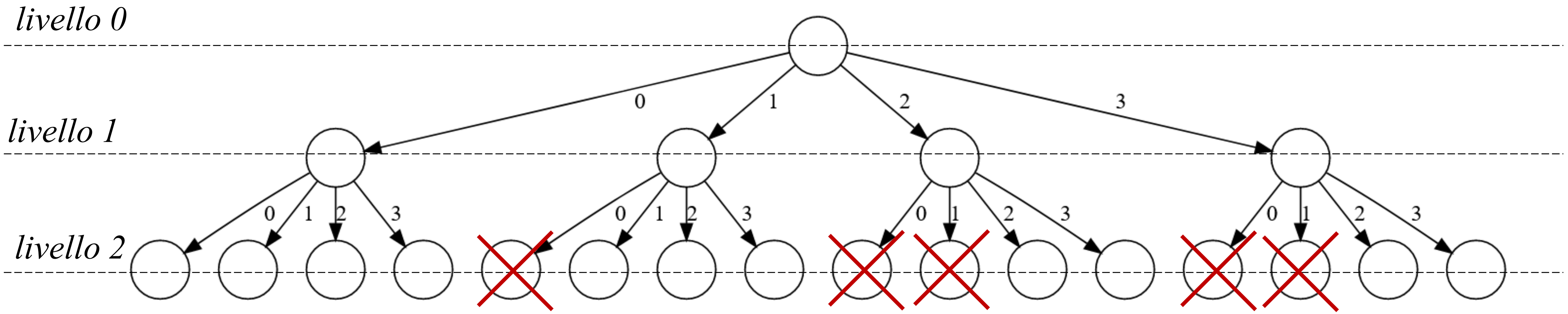
Soluzioni Non Ammissibili (vincoli)

- L’esplorazione di questo albero può produrre soluzioni ripetute.
- Il testo non fa distinzione tra i ragazzi, quindi, mettere il primo nell’ombrellone di posto 0 e il secondo nell’ombrellone di posto 2 equivale a mettere il primo nell’ombrellone di posto 2 e il secondo in quello di posto 0.
- Solo una delle soluzioni equivalenti deve essere considerata.



Soluzioni Non Ammissibili (vincoli)

- Anche questa eventualità veniva scartata a priori con la modellazione precedente.
- In questo caso devo dire che se al passo i ho fatto la scelta x al passo successivo $i + 1$ potrò scegliere solo i rami corrispondenti a scelte $y \geq x$



Soluzioni Non Ammissibili (vincoli)

- Applicando tutte le potature descritte otteniamo l'albero (potato) qui riportato.
- In sostanza, l'applicazione di tutti i vincoli descritti si traduce in: *se al passo i hai fatto la scelta x, al passo $i+1$ potrai fare solo le scelte $y \geq x+2$*