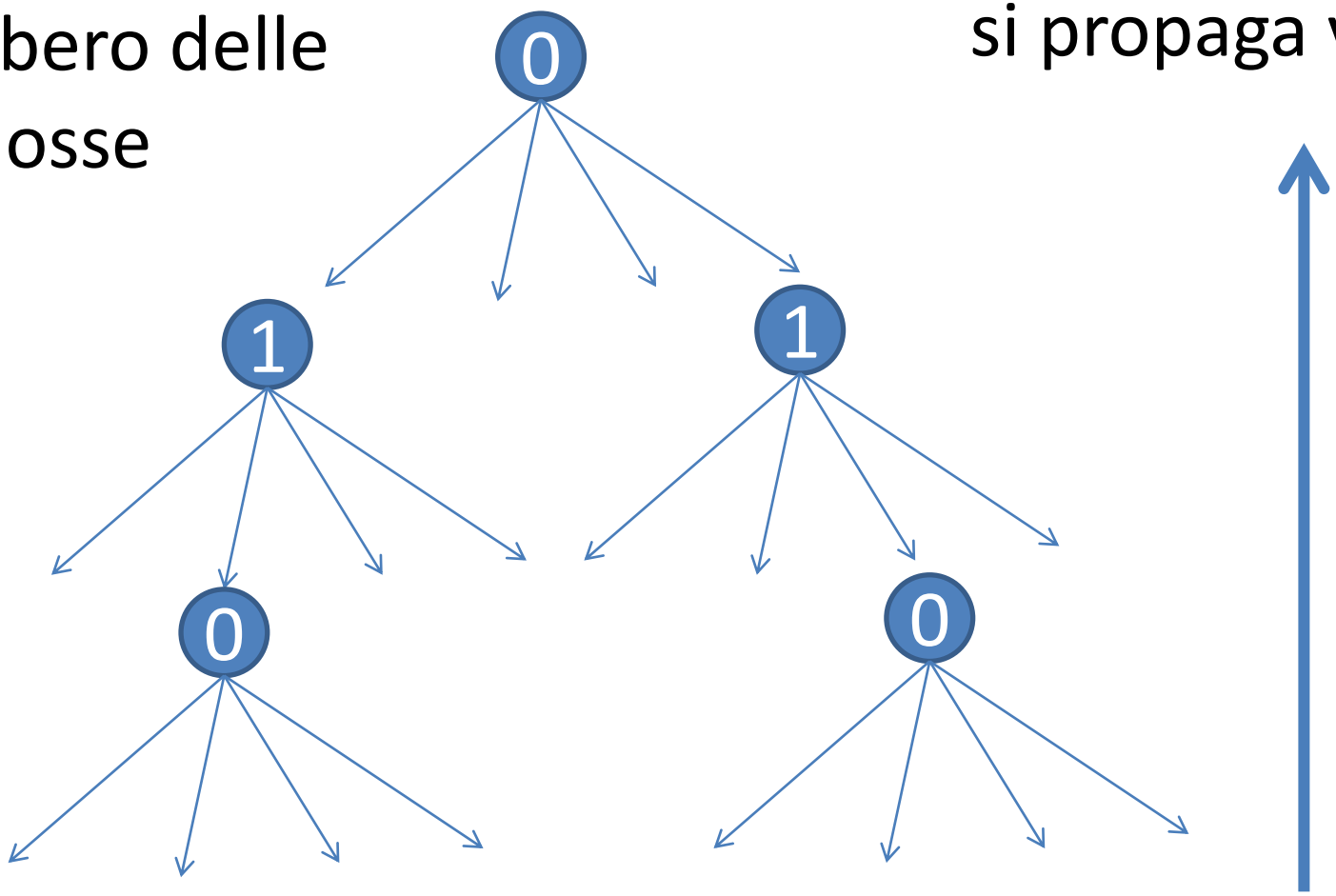
progetto Bantumi

consegna libera

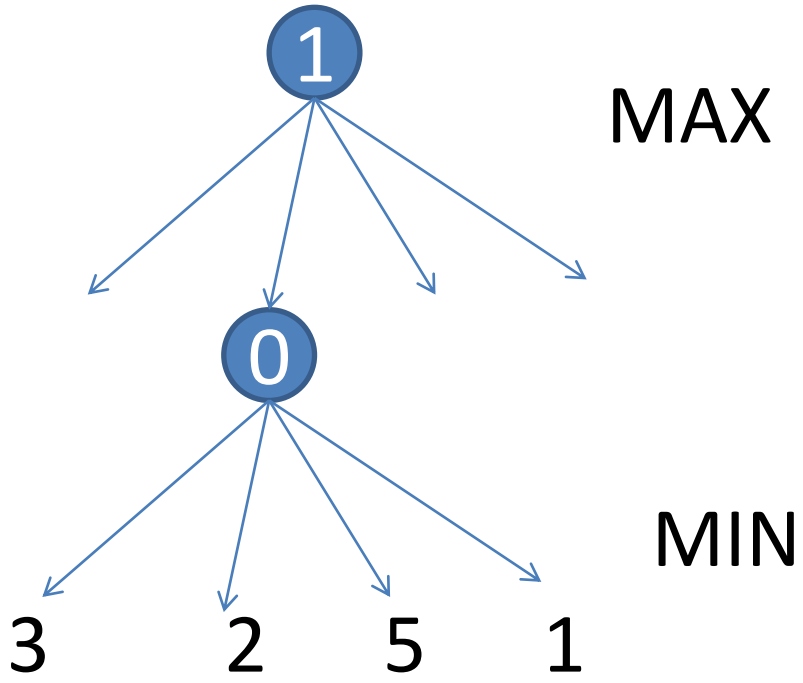
vale 2 punti extra

albero delle mosse

si propaga voto

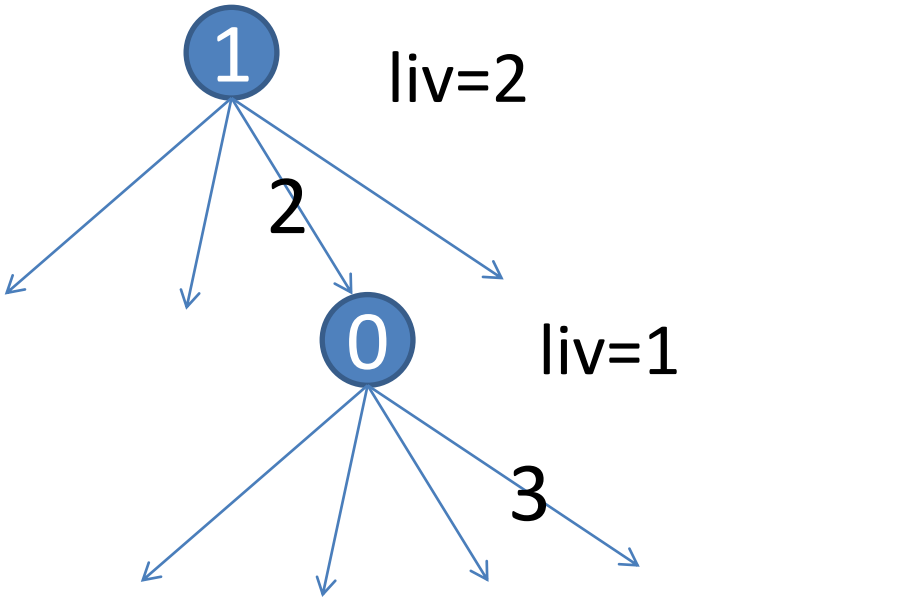


valutazione statica delle configurazioni raggiunte



valutazione statica fornisce dei VOTI di 1 per le configurazioni raggiunte

prendiamo un cammino



lo descriviamo come sequenza di triple:
(2,1,2) (1, 0,3) (0,1,0) dove la prima componente è il livello, la seconda componente è il giocatore e la terza è la mossa. La terza tripla ha mossa =0 perché è il risultato della valutazione statica

Si richiede di scrivere un programma che legge da “input” una configurazione del gioco, il player che deve giocare (0/1) e il livello liv da usare nella ricorsione (0,2,4,...) e produce (da sinistra a destra) tutte le sequenze di triple che corrispondono ai cammini radice-foglia dell’albero delle mosse:
Si osservi che non basta calcolare tutte le sequenze di liv+1 triple che alternano i giocatori. Infatti è necessario:
i) considerare solo mosse possibili, cioè la buca scelta per la mossa non deve essere vuota;
ii) riconoscere le configurazioni di fine gioco che devono essere foglie dell’albero;
iii) riconoscere che una mossa causa il fatto che lo stesso giocatore abbia diritto di rigiocare. In questo caso quindi nella tripla successiva il giocatore non deve cambiare.

Insomma nel discendere ricorsivamente nell’albero delle mosse, si deve effettivamente effettuare le mosse indicate nel cammino.
Per ogni nuova scelta di mosse è necessario fare una copia della configurazione raggiunta fin lì, altrimenti le diverse mosse da tentare (separatamente) si sovrapporrebbero tra loro.
Per finire, sebbene la stampa delle triple sia richiesta secondo l’ordine che va dalla radice dell’albero verso una foglia, è necessario farla quando si è su una foglia, perché solo sulle foglie si conosce il cammino che ha portato fino a loro

per risolvere l'esercizio si consiglia di usare la soluzione all'esercizio 4 del 17/2 che potete trovare allegata ad un mio post sull'elearning del corso.
In particolare in quel programma c'è una funzione che "fa" una data mossa, restituendo anche l'informazione se il giocatore ha o meno diritto di giocare di nuovo e ci sono anche (nel main e quindi vanno incapsulate in una funzione) le istruzioni per stabilire se si è arrivati ad una configurazione di fine-partita.

Queste funzioni sono molto utili per risolvere questo esercizio.

Visto che l'esercizio è piuttosto complicato, la sua consegna non è richiesta per partecipare al II compitino e nemmeno ai successivi esami. Insomma la consegna è volontaria e anche il tempo di consegna non è fissato.
La consegna di una soluzione corretta garantisce 2 punti di maggiorazione rispetto al voto dello scritto. Alcuni di coloro che consegneranno l'esercizio verranno scelti dal docente per una verifica orale del fatto che sappiano cosa hanno consegnato.

esempio: si consideri il seguente input, dove i primi 14 interi sono la configurazione iniziale del gioco e poi viene specificato il giocatore (0) e il livello (2):

0 0 9 0 0 5 10
4 0 0 0 0 10 6
0 2

l'output da produrre in questo caso sarebbe:

(2,0,2)(1,1,0)(0,0,0)
(2,0,2)(1,1,1)(0,0,0)
(2,0,2)(1,1,2)(0,0,0)
(2,0,2)(1,1,3)(0,0,0)
(2,0,2)(1,1,5)(0,0,0)
(2,0,5)

si osservi che l'ultima riga corrisponde alla mossa 5 del giocatore 0 che porta alla immediata fine del gioco. Le prime 4 righe corrispondono alle diverse scelte (mossa 0, 1, 2, 3, e 5) che il giocatore 1 può fare in risposta alla mossa 2 del giocatore 0. Si osservi che dopo la mossa 2 del giocatore 0, la buca 4 del giocatore 1 sarebbe vuota e per questo non c'è la tripla (1,1,4) tra quelle in seconda posizione.

Ancora di più ....

In caso qualcuno voglia fare anche una funzione di valutazione delle configurazioni raggiunte alla frontiera dell'albero delle mosse e produca quindi un programma capace di giocare il Bantumi in modo autonomo, possiamo organizzare un torneo facendo giocare tra loro funzioni di studenti diversi (e anche la mia) per vedere quale proposta si rivela migliore.

L'unica richiesta è che lo studente consegni un file con una funzione con questo prototipo:
int meglio_mossa(int B[][7], int player, int livello)

capace quindi di comunicare la prossima mossa che player deve fare nella configurazione rappresentata da B. Il parametro livello è profondità dell'albero delle mosse che va costruito. Significa che la vostra meglio_mossa deve poter costruire (almeno in linea di principio) alberi delle mosse di profondità qualsiasi.