sabato 13 aprile 2019

Tablut —idee

MinMax

- Ottimizzare min max:

si effettua una prima "pre-ricerca" molto superficiale per ottenere un primo albero già parzialmente ordinato, che la ricerca vera e propria si occuperà di approfondire e completare fino alla profondità stabilita.
Si possono migliorare le prestazioni senza sacrificare l'accuratezza dei risultati usando euristiche di ordinamento per ricercare subito le parti dell'albero che più probabilmente forzeranno subito dei tagli alfa-beta: per esempio negli scacchi si esaminano per prime le mosse che prendono dei pezzi, o che hanno raggiunto dei punteggi molto alti nella prima ricerca superficiale.Un'altra ottimizzazione euristica molto comune ed economica è l'euristica killer, per cui l'ultima mossa che ha provocato un taglio beta allo stesso livello nell'albero viene sempre ricercata per prima; questa ottimizzazione si può generalizzare in un insieme di tabelle di confutazione.
La ricerca alfa-beta può essere ulteriormente accelerata considerando una finestra di ricerca (la differenza **α - β**) molto stretta, di solito con un valore ipotizzato in base all'esperienza; questa tecnica è nota come *aspiration search*. Nel caso estremo, si compie la ricerca con **α** = **β**, ottenendo la tecnica nota come *zero-window search*, *null-window search*, o *scout search*. Questa tecnica è particolarmente efficace nei finali di partita, quando si cerca lo scaccomatto. Se una aspiration search fallisce, è immediato sapere se l'intervallo scelto era troppo alto o troppo basso, ottenendo una nuova ipotesi di intervallo alfa-beta per una eventuale nuova ricerca sulla stessa posizione.

—Fonte Wikipedia

Link utili:

- https://javarevisited.blogspot.com/2016/07/binary-tree-preorder-traversal-in-java-using-recursion-iteration-example.html
- https://javarevisited.blogspot.com/2016/08/inorder-traversal-of-binary-tree-in-java-recursion-iteration-example.html