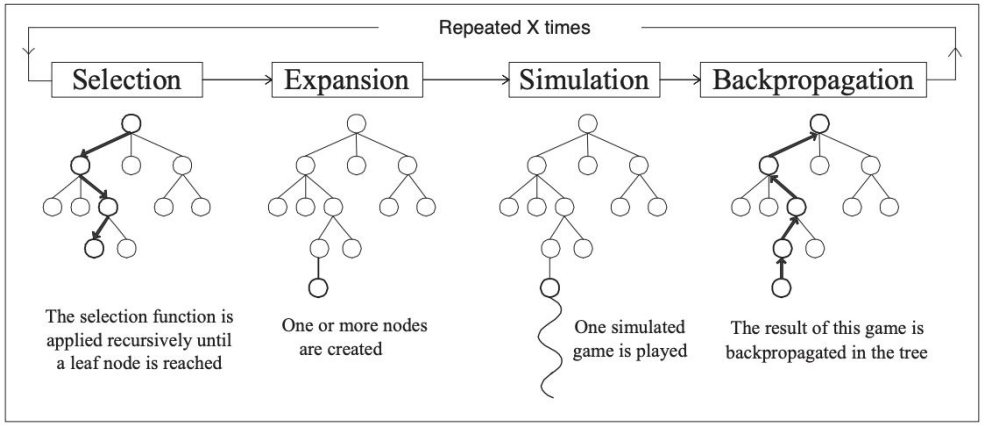
AIMAteurs

(AKA Melanzanina Iterative Deepening)

Tablut Challenge 2024-25

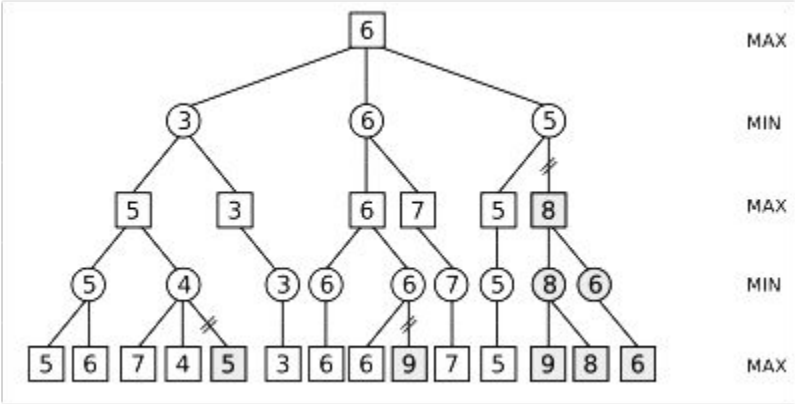
SCELTA DELL'ALGORITMO

- Spazio degli stati esteso → necessità di usare un algoritmo ottimizzato (stack overflow se si usa MinMax puro)
- Scelta iniziale = **Monte Carlo Tree Search**, algoritmo alla base di AlphaGo
  - Problemi riscontrati: troppe mosse svantaggiose tra quelle possibili fin da inizio partita, algoritmo penalizzato dal basso tempo a disposizione



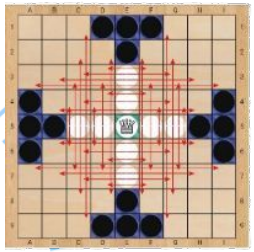
SCELTA DELL'ALGORITMO

- Scelta finale = **Iterative Deepening Alpha-Beta Search**
  - tutti i vantaggi del MinMax
  - ottimizzazione grazie ai tagli Alpha-Beta
  - profondità di esplorazione adattata in base al tempo disponibile
- Implementazione semplice grazie alla libreria AIMA

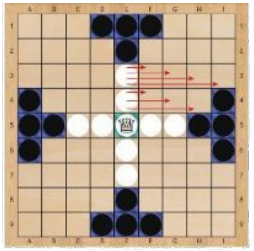


Simmetrie

- A parità di euristica e algoritmo, vince la modellazione più efficiente dei metodi **getResult()** e **getAction()** → per massimizzare l'efficienza si è provato a sfruttare la **struttura simmetrica del tabellone** e un meccanismo di **caching**
- Ogni stato viene ricondotto a una forma "canonica", salvandosi la rotazione o simmetria con cui lo abbiamo incontrato
- Se è già stato analizzato uno stato simmetrico in qualche modo a quello attuale, ci si riconduce alle mosse già calcolate
- Una volta valutato il risultato di una mossa questo viene convertito a sua volta nella sua forma canonica e salvato nelle cache dopo aver controllato se si ricade nel pareggio (che invece dipende dalla rotazione e va trattato di conseguenza)

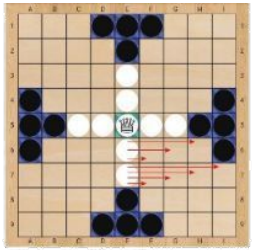


=



CANONICA

+



SIMM. ORIZZ.

+

SIMM. VERT.

+

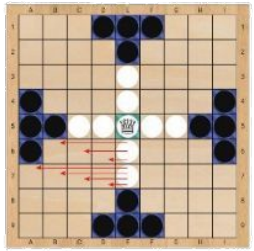
SIMM. DIAG. 1

SIMM. DIAG. 2

+

ROTAZ. 90°

+



ROTAZ. 180°

+

ROTAZ. 270°

Simmetrie

- Conversione migliorabile, **ancora costosa** (perché vanno valutate tutte le simmetrie del tabellone orientato e sceglierne una canonica), ma migliore rispetto a salvare come "canonica" la prima rotazione trovata
  - se so quale rotazione mi porta alla forma canonica, posso fare direttamente quella trasformazione
  - altrimenti, ogni volta dovrei applicare tutte le rotazioni e simmetrie e confrontare con gli stati già visti
- Come già detto, nel controllo delle condizioni di pareggio questo metodo non è il più efficace

Testing

- Il player ha sfidato alcuni giocatori degli anni passati il cui codice è stato trovato su GitHub, in particolare il player del team Gionnino9000, Tavoletta
- A parità di euristica e algoritmo, i nodi valutati sono molto ridotti
- Da qui deriva il nome del player rimasto in alcune print, ovvero TavolettaSlayer :)