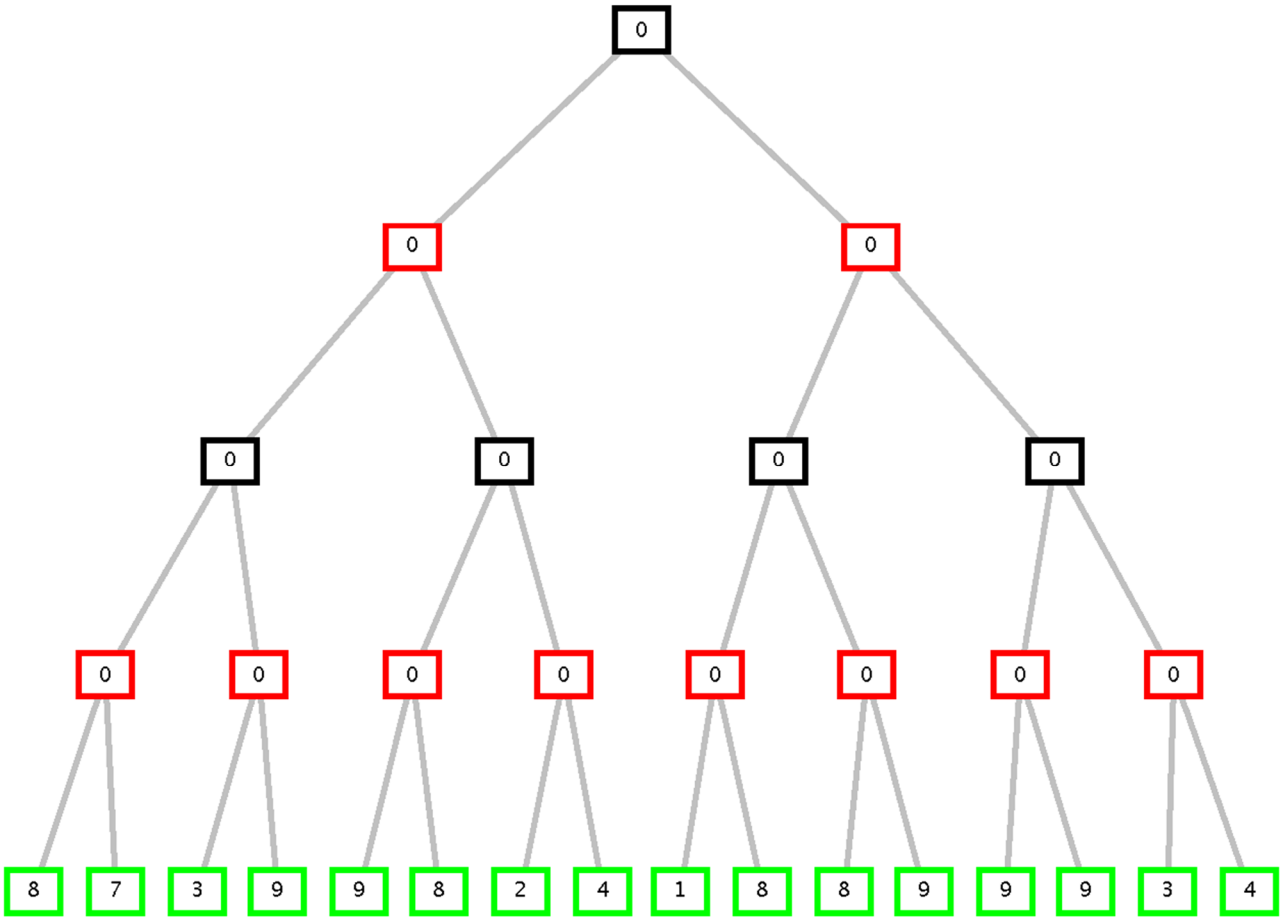
FONDAMENTI DI INTELLIGENZA ARTIFICIALE
13 Settembre 2018 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (punti 6)
Modellare in logica del I ordine la seguente frase in linguaggio naturale:

Se il Segretario propone un Membro del Consiglio Direttivo, allora il Presidente dell'Associazione lo nomina

Sapendo che Giovanni è Presidente dell'Associazione, Mario è Segretario, che Luca è stato proposto Membro del Consiglio Direttivo da Mario e che Giovanni non ha nominato membro del Consiglio Direttivo Luca, modellare tali fatti, e aggiungendoli alle formule precedenti dimostrare, tramite il principio di risoluzione, che la teoria – trasformata in clausole – è contraddittoria.
Si utilizzi un linguaggio logico con i seguenti predicati:
`presidente(X)`: X è presidente dell'Associazione;
`segretario(X)`: X è segretario dell'Associazione;
`propone(X,Y)`: X propone come membro del Direttivo Y;
`nomina(X,Y)`: X nomina come membro del Direttivo Y.

Esercizio 2 (punti 5)
Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore (*MAX*). Si mostri come l'algoritmo *min-max* e l'algoritmo *alfa-beta* risolvono il problema e la mossa selezionata dal giocatore.



Esercizio 3 (punti 6)
Dato il seguente programma Prolog:

```
couple([A,B]) :- A==B, !.
couple([A,B|_]) :- A==B, !.
couple([_|Tail]) :- couple(Tail).
```

dove il predicato `couple(List)` verifica se nella lista `List` esistono due elementi adiacenti uguali, si disegni l'albero SLD del goal:

```
?- couple([4, a, 2, 2, b]).
```

Esercizio 4 (punti 5)
Si scriva un predicato Prolog `included(L1, L2)` che date le liste di atomi `L1` e `L2` verifica se gli elementi della prima sono tutti anche nella seconda. Si definiscano tutti i predicati che si utilizzano.
Esempio:

```
?- included([a,b,c],[a,b,c]).
Yes
?- included([a,b,c],[b,c,a]).
Yes
?- included([a,b,c],[a,b,c,a]).
Yes
?- included([a,b,c],[1,2,a,b]).
No
```

Esercizio 5 (punti 8)
Si consideri il problema delle 4 Regine e lo si modelli mediante un CSP. Si utilizzino 4 variabili (per i=1..4) denominate RCi, ovvero Regina in Colonna i; il valore di ciascuna variabile è la riga in cui è posizionata la regina.
Si disegni poi il relativo constraint graph in cui tutti i vincoli che coinvolgono le stesse variabili sono rappresentati con un singolo arco etichettato dall'"and" dei vincoli componenti.
Si consideri poi il caso in cui esista un ulteriore vincolo unario che vincoli la regina in colonna 1 ad essere posizionata in riga 2 e si mostri il risultato dopo l'applicazione della Node e poi della Arc-Consistenza.
Si determini inoltre la soluzione se esiste.
Si ripeta poi lo stesso procedimento, ma partendo da un vincolo unario che vincoli la regina in colonna 1 ad essere posizionata in riga 1.
Si determini anche in questo caso la soluzione se esiste.

Esercizio 6 (punti 2)
Si descrivano la ricerca depth-first (in profondità) e la ricerca iterative-deepening (ad approfondimento iterativo) e le loro proprietà (completezza, complessità in tempo e in spazio, ottimalità).

13 Settembre 2018 – Soluzioni

Esercizio 1

∀X ∀Y ∀Z presidente(X), propone(Z,Y), segretario(Z) ⇒ nomina(X,Y).
presidente(giovanni).
segretario(mario).
propone(mario,luca).
not nomina(giovanni,luca).

Trasformazione in Clausole:

1: ¬presidente(X) ∨ ¬propone(Z,Y) ∨ ¬segretario(Z) ∨ nomina(X,Y).
2: presidente(giovanni).
3: segretario(mario).
4: propone(mario,luca).
5: ¬nomina(giovanni,luca).

Risoluzione:

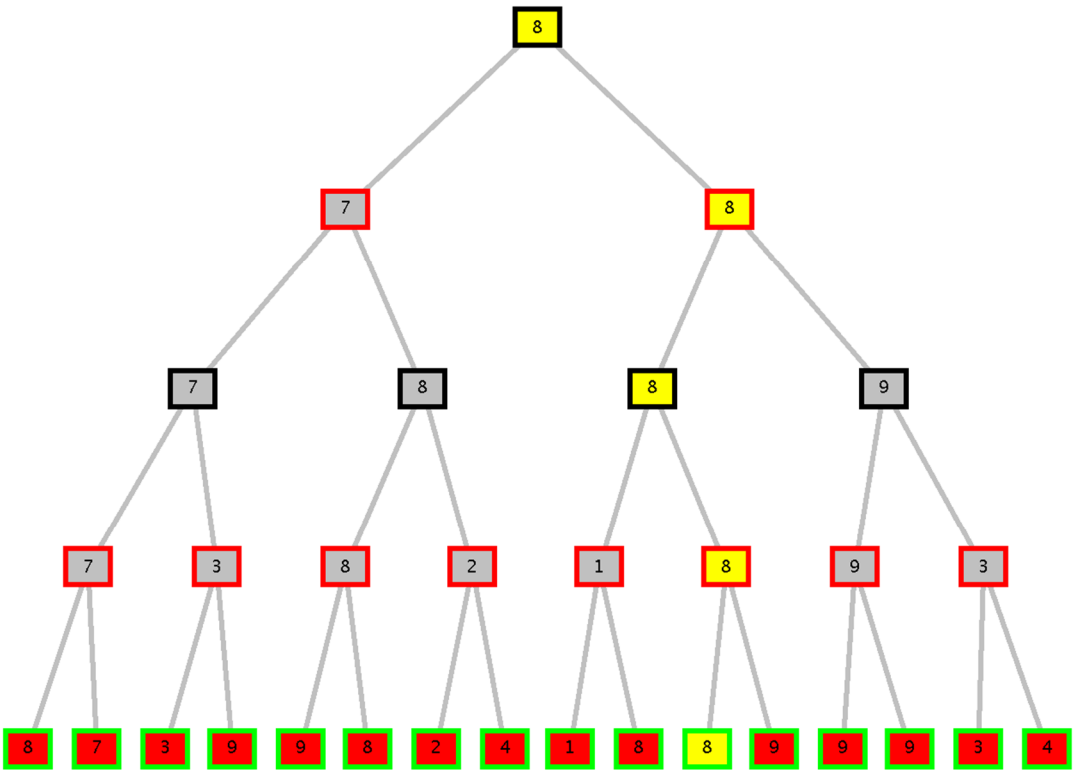
Da 1)+2)

6: 1+2: ¬propone(Z,Y) ∨ ¬segretario(Z) ∨ nomina(giovanni,Y).
7: 6+3: ¬propone(mario,Y) ∨ nomina(giovanni,Y).
8:7+4: nomina(giovanni,luca).

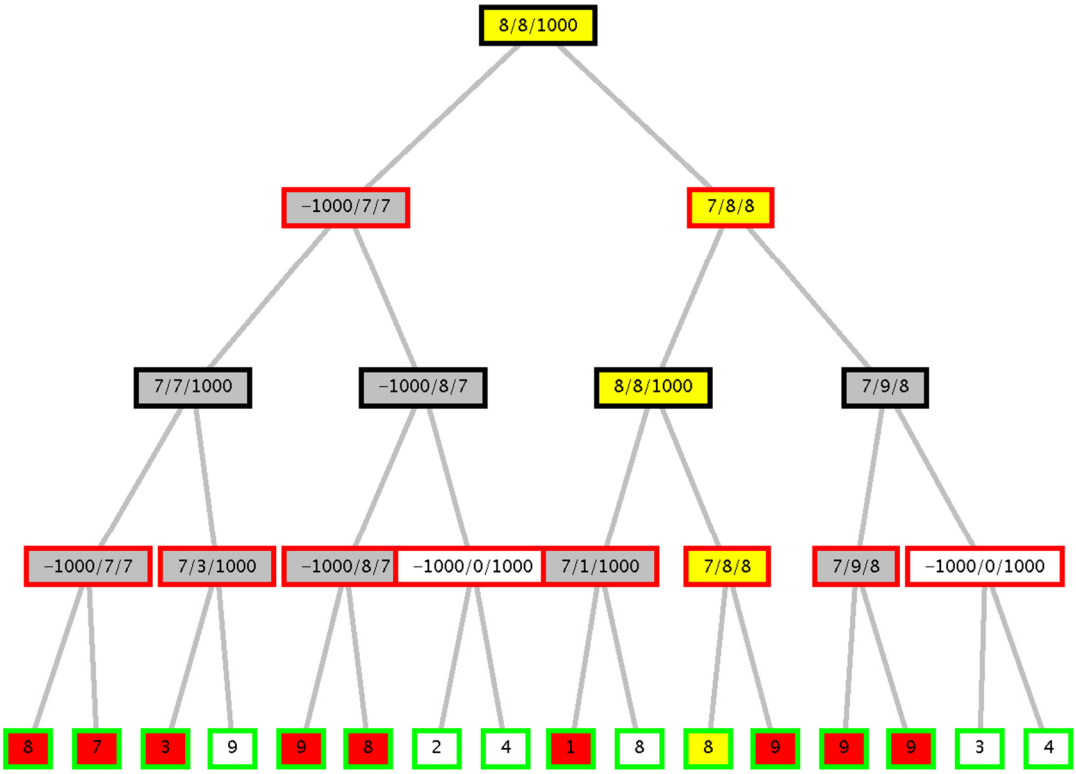
Da 5+8 segue contraddizione.

Esercizio 2

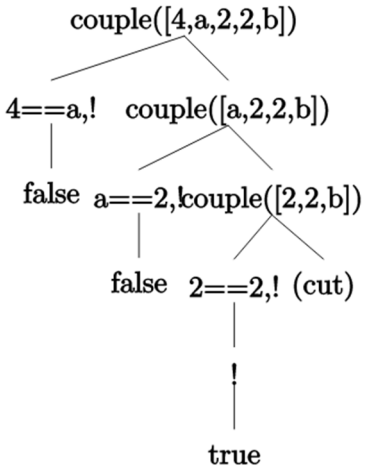
Min-Max:



Alfa-beta: I nodi che portano alla soluzione sono in giallo, quelli tagliati in bianco.



Esercizio 3



Esercizio 4

```
included([],_).
included([H|T], L2) :-
     member(H, L2),
     included(T,L2).
```

Esercizio 5

Domini:
dom(RC1)={1,2,3,4}
dom(RC2)={1,2,3,4}
dom(RC3)={1,2,3,4}
dom(RC4)={1,2,3,4}

Constraints:

{RC1≠RC2≠RC3≠RC4)

ABS(RC1-RC2)≠1

ABS(RC1-RC3)≠2

ABS(RC1-RC4)≠3

ABS(RC2-RC3)≠1

ABS(RC2-RC4)≠2

ABS(RC3-RC4)≠1

Applicazione dell'algoritmo di Arc-Consistency AC3 (si vedano le slide del corso).

Caso 1) RISULTATO FINALE:
dom(RC1)={2}
dom(RC2)={4}
dom(RC3)={1}
dom(RC4)={3}
Esiste un'unica soluzione che si può verificare rispetta tutti i vincoli. RC1 = 2; RC2= 4; RC3=1 ; RC4 = 3.

Caso 2) RISULTATO FINALE:
dom(RC1)={1}
dom(RC2)={3,4}
dom(RC3)={2}
dom(RC4)={ }
Non esiste alcuna soluzione perché si vuota il dominio di C4.

Constraint graph di partenza: