FONDAMENTI DI INTELLIGENZA ARTIFICIALE

13 Febbraio 2024 - Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)

Si formalizzino le seguenti frasi in logica dei predicati del I ordine:

1. *Tutti i soci dell'associazione a1 sono ingegneri o artisti* (or non esclusivo)
2. *Tutti i soci dell'associazione a2 non sono ingegneri*
3. *Giovanni non è un artista*
4. *Giovanni è socio dell'associazione a2*
5. *Per ogni ingegnere esiste un artista che lo ammira*.

Le si trasformi in clausole e si usi la risoluzione per dimostrare che (query):

Query: *Giovanni non è un socio di a1.*

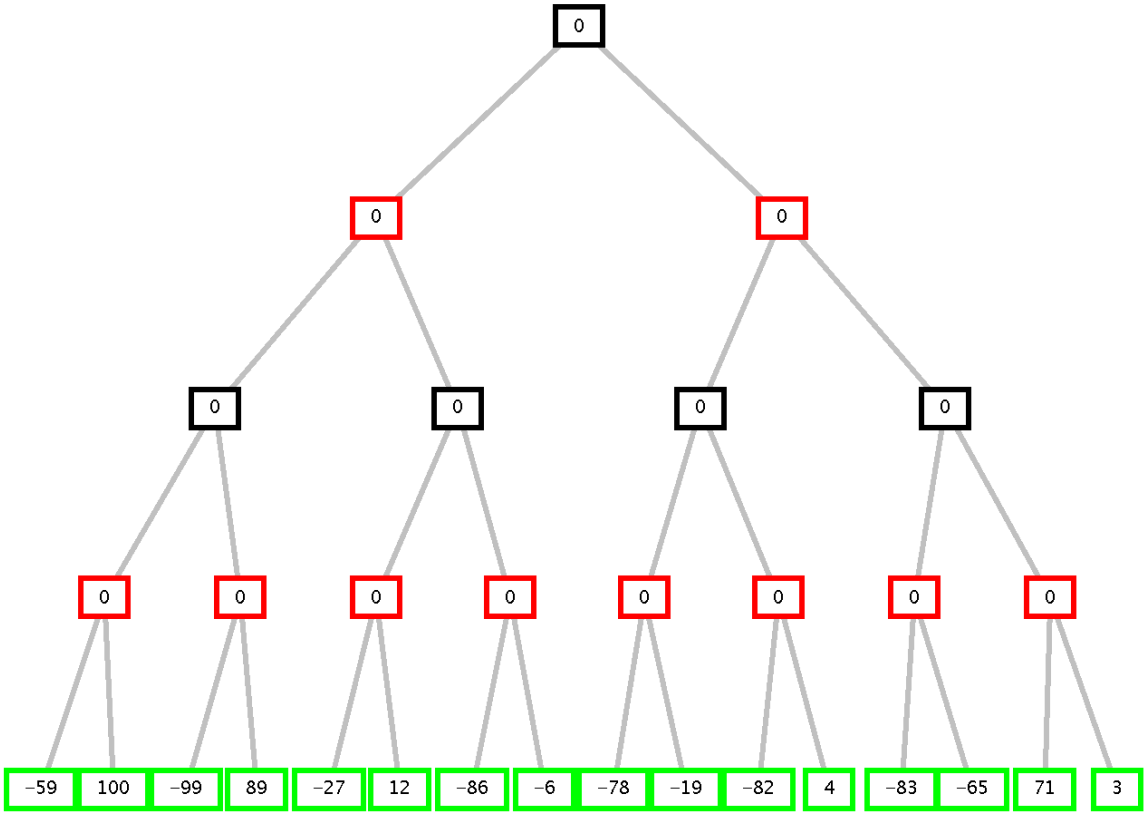
Si usino i predicati:

socio_a1(X), X è socio dell'associazione a1
socio_a2(X), X è socio dell'associazione a2
artista(X), X è un artista
ingegnere(X), X è un ingegnere
ammira(X,Y), X ammira Y

e la costante: giovanni.

Esercizio 2 (5 punti)

Si consideri il seguente albero di gioco in cui il primo giocatore è MAX.



a) Si indichi come l'algoritmo min-max risolve il problema indicando il valore con cui viene etichettato il nodo iniziale e la mossa selezionata dal primo giocatore (arco a sinistra o a destra).
b) Si mostrino poi i tagli che l'algoritmo alfa-beta consente, indicando gli archi che verranno tagliati.

Esercizio 3 (5 punti)

Si scriva un predicato Prolog **selezMax(X,L,Lout)** che, dato un numero intero **X** e una lista di numeri interi **L**, ha successo con **Lout** lista che contiene tutte le occorrenze di numeri in **L** maggiori di **X**.

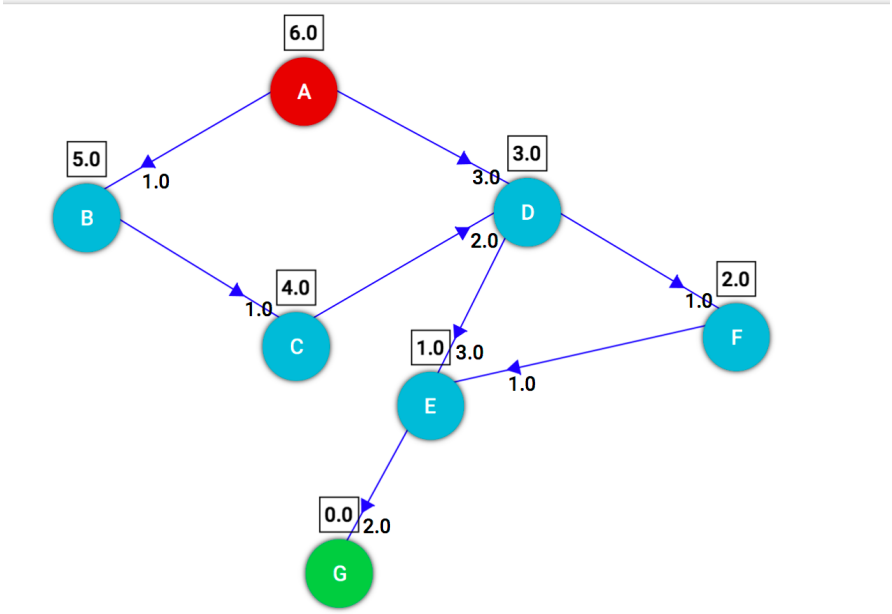
Esempi:

?- selezMax(1, [1,4,5,4,1,1], L). yes L=[4,5,4]
?- selezMax (2, [1,1,1], L). yes L=[]
?- selezMax(5, [4,1,1], [4]). no

Esercizio 4 (5 punti)
Dato il seguente programma Prolog **lastLists(L1,L2)**:
lastLists([],[]).
lastLists([[]|T], T1):- !,lastLists(T,T1).
lastLists([X|T], [Z|T1]):-last(Z,X), lastLists(T,T1).

last(X,[X]):-!.
last(X,[_|T]):-last(X,T).
che, data una lista di liste di interi **L1**, restituisce in uscita una lista **L2** contenente tutti gli elementi in ultima posizione delle liste che compongono **L1** (se una lista in **L1** è vuota non verrà considerata), si mostri l'albero SLD generato dal goal:
?- lastLists([[3],[4,9]],[3,9]).
indicando eventuali rami di fallimento e quelli tagliati da cut.

Esercizio 5 (6 punti)
Si consideri il seguente grafo, dove A è il nodo iniziale e G il nodo goal, e il numero associato agli archi è il costo dell'operatore per andare dal nodo di partenza al nodo di arrivo dell'arco. Vicino ad ogni nodo, in un quadrato, è indicata inoltre la stima euristica della sua distanza dal nodo goal G:



Si applichi la ricerca **A*** su alberi (non tenendo quindi traccia dei nodi già visitati che non vengono automaticamente eliminati) **disegnando l'albero generato dinamicamente.** In caso di non determinismo si selezionino i nodi da espandere secondo l'ordine alfabetico. Si indichino:

- i nodi espansi nell'ordine di espansione;
- i nodi sulla strada della soluzione e il costo della soluzione;

Si indichi per la ricerca **A*** la condizione sulla stima euristica h(n) che garantisce l'ottimalità di A* su alberi e se è soddisfatta in questo caso.

Esercizio 6 (5 punti)
Dopo avere brevemente introdotto l'algoritmo di Arc-Consistency, si disegni il grafo per questo CSP e si faccia vedere l'applicazione di AC sul problema in esame.
A,B,C,D::[1, 2, 3, 4, 5, 6, 7]
A<B-3
A!=C
D=B+2
C<D-4

13 Febbraio 2024 - Soluzioni

Esercizio 1

Rappresentazione in FOL:

1. ∀ X socio_a1(X) -> (ingegnere(X) or artista(X)).
2. ∀ X socio_a2(X) -> not ingegnere(X).
3. not artista(giovanni).
4. socio_a2(giovanni).
5. ∀ X ingegnere(X) -> ∃Y artista(Y) and ammira(Y,X)

Query: not socio_a1(giovanni).

Clausole:

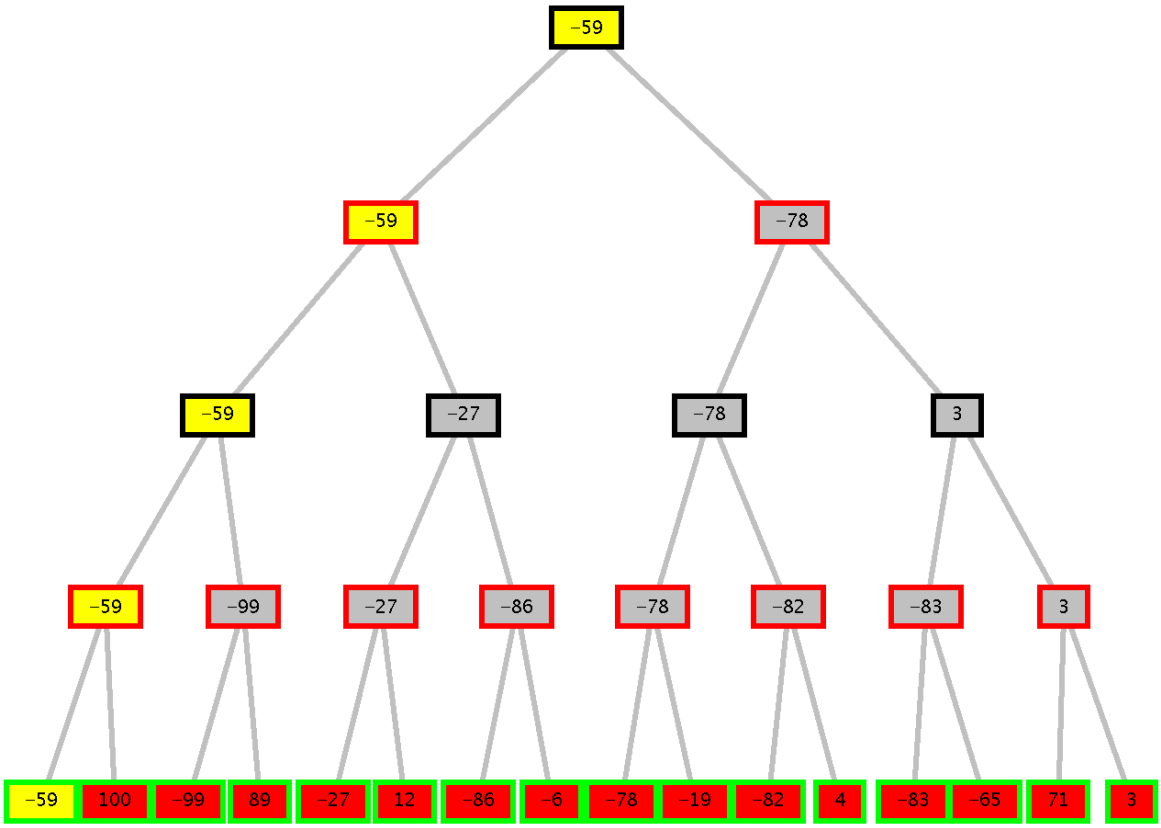
C1: not socio_a1(X) or ingegnere(X) or artista(X).
C2: not socio_a2(X) or not ingegnere(X).
C3: not artista(giovanni).
C4: socio_a2(giovanni).
C5: ∀X ∃Y (not ingegnere(X) or (artista(Y) and ammira (Y,X)))
C5a: not ingegnere(X) or artista(p(X)) *funzione di SKOLEM*
C5b: not ingegnere(X) or ammira(p(X), X) *funzione di SKOLEM*
QueryNeg: socio_a1(giovanni).

Risoluzione:

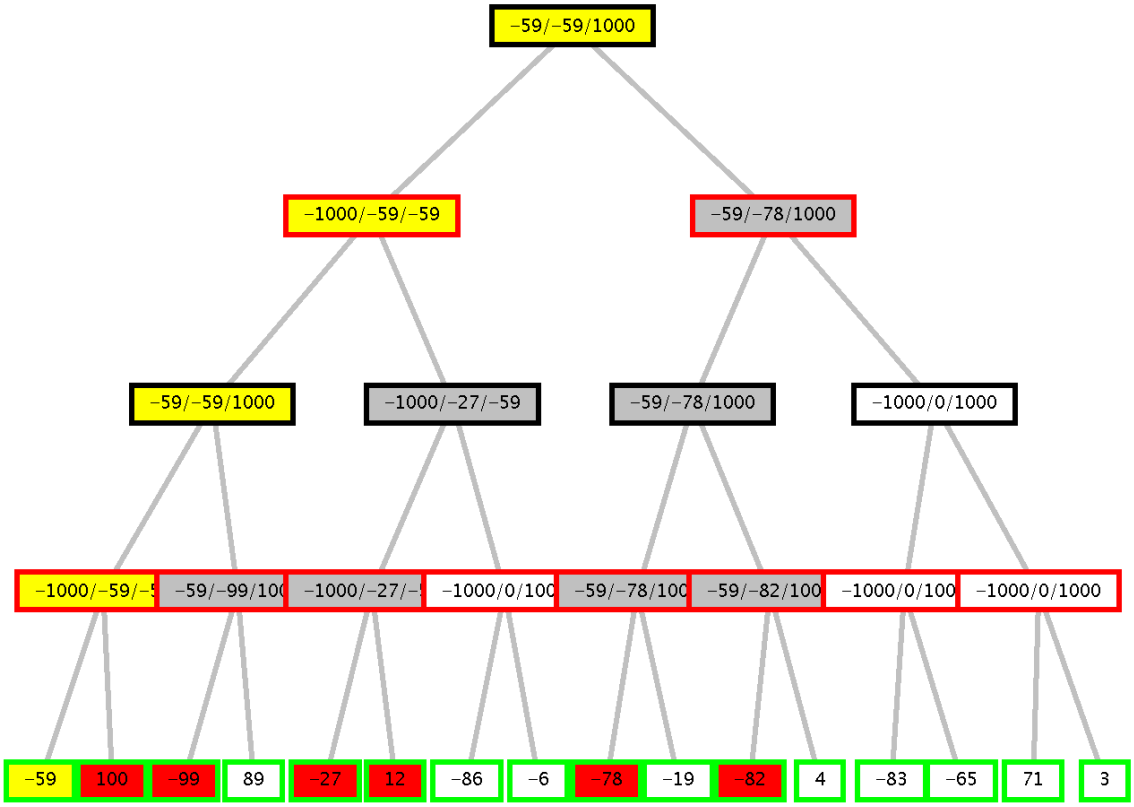
QueryNeg + C1 =	C6: ingegnere(giovanni) or artista(giovanni).	
C6 + C3 =	C7: ingegnere(giovanni)	
C7 + C2 =	C8 : not socio_a2(giovanni)	
C8 + C4 =	contraddizione!	*dimostrato*

Esercizio 2

Min max:



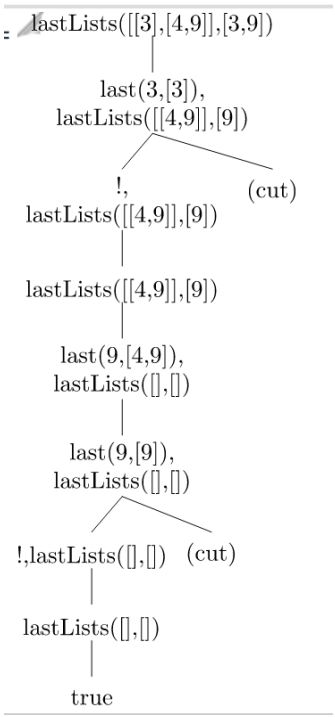
Alfa Beta:



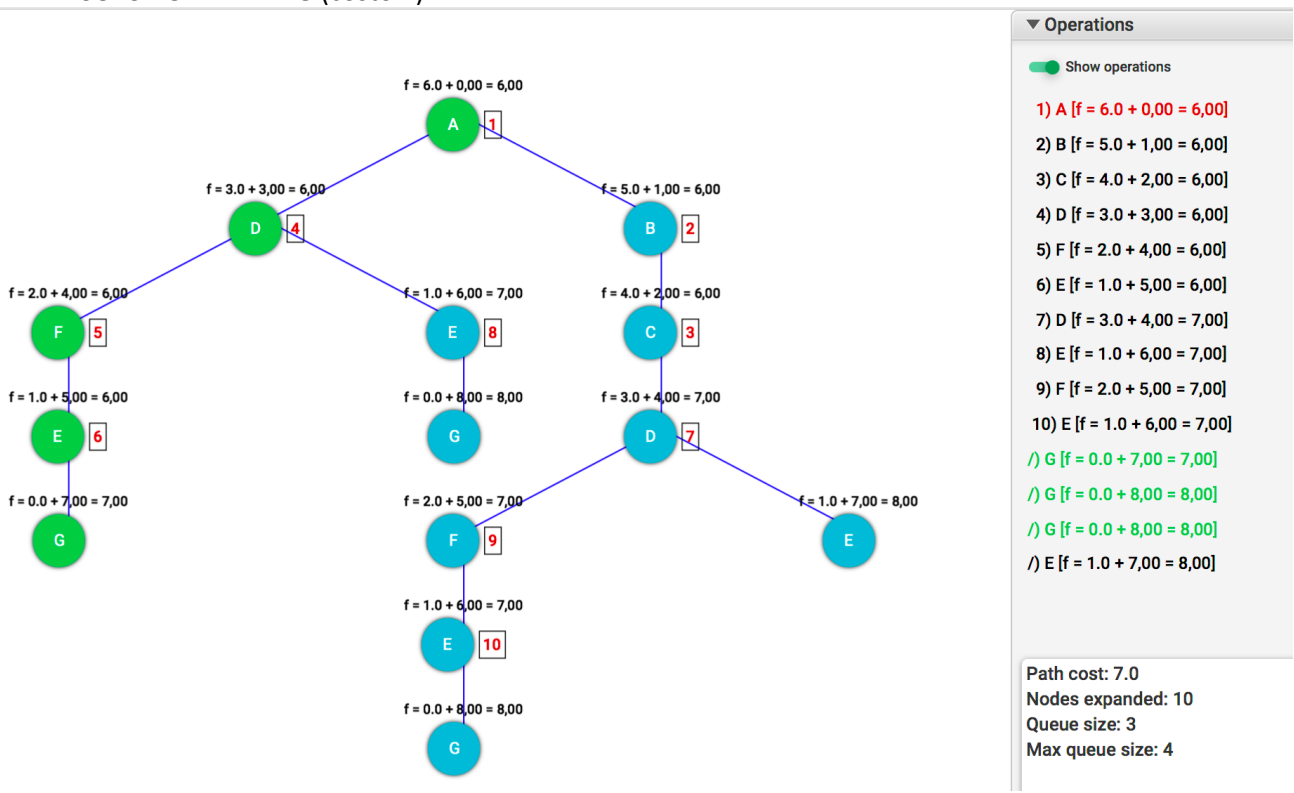
Esercizio 3

```
selezMax(_,[],[]):-!.
selezMax(X,[Y|T], [Y|L]) :- Y > X,!,selezMax(X,T,L).
selezMax(X,[_|T], L) :- selezMax(X,T,L).
```

Esercizio 4



Esercizio 5
Con A*:
NODI ESPANSI: ABCDFEDEFEG
STRADA DELLA SOLUZIONE: ADFEG (costo 7)



La condizione sulla funzione euristica stimata h* (n) che garantisce l'ottimalità della ricerca è la condizione di ammissibilità che deve valere per ogni nodo dell'albero e che è verificata se la h*(n) è ottimista cioè h*(n) <= h(n). Tale condizione è soddisfatta in questo caso.

Esercizio 6 Vedi slide del corso per spiegare Arc-Consistency.

	A	**B**	**C**	**D**
1 iteraz.	[1, 2, 3, 4, 5, 6, 7]	[1, 2, 3, 4, 5, 6, 7]	[1, 2, 3, 4, 5, 6, 7]	[1, 2, 3, 4, 5, 6, 7]
A<B-3	[1,2,3]			
B>A+3		[5, 6, 7]		
A!=C				
C!=A				
D=B+2				[7]
B=D-2		[5]		
C<D-4			[1,2]	
D>C+4				
2 iteraz.	**[1, 2, 3]**	**[5]**	**[1, 2]**	**[7]**
A<B-3	[1]			
B>A+3				
A!=C				
C!=A			[2]	
D=B+2				
B=D-2				
C<D-4				
D>C+4				
3 iteraz.	**[1]**	**[5]**	**[2]**	**[7]**

Nessuna ulteriore cancellazione dai domini. L'assegnazione dell'unico valore a ogni variabile è la soluzione.