FONDAMENTI DI INTELLIGENZA ARTIFICIALE

15 Giugno 2023– Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)

a) Si modellino in logica dei predicati del I ordine le seguenti frasi:

Chiunque Giorgia ammira è affidabile.
Qualsiasi sportivo che non si impegna non raggiunge gli obiettivi
Esiste uno sportivo che non si allena.
Qualsiasi sportivo che non si allena non si impegna.
Chiunque non raggiunge gli obiettivi non è affidabile.

b) Le si trasformi in logica a clausole e si dimostri, applicando il principio di risoluzione, che:

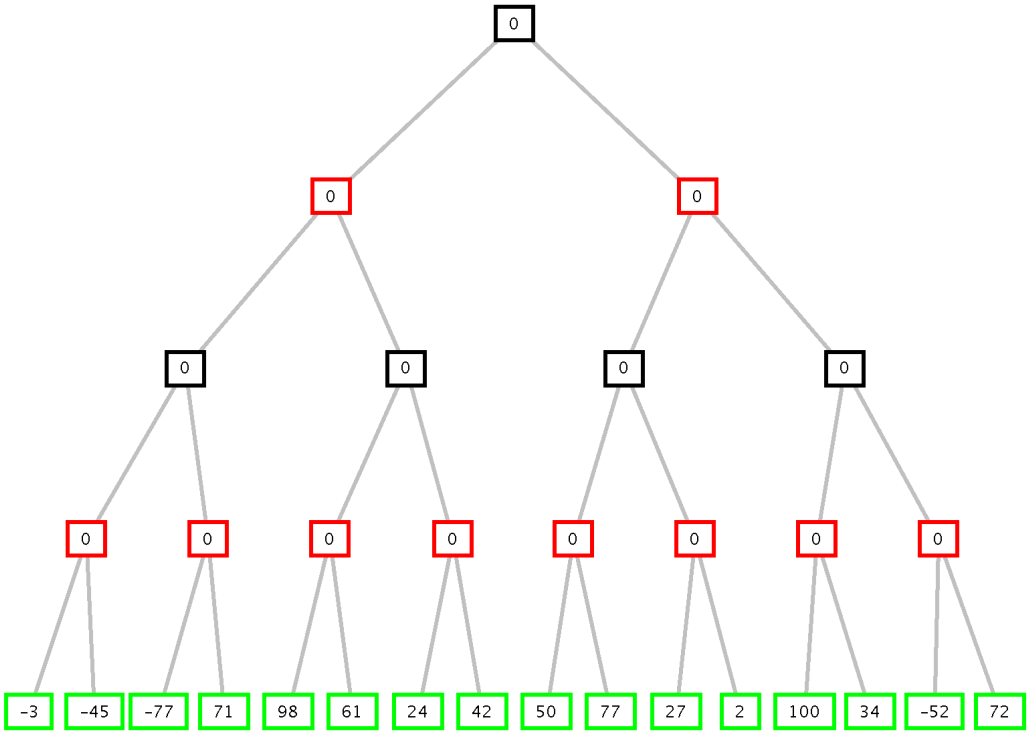
Esiste qualcuno che Giorgia non ammira

Si utilizzino I seguenti predicati:

ammira(X,Y): X ammira Y;
affidabile(X): X è affidabile
sportivo(X): X è sportivo;
impegna(X): X si impegna
obiettivi(X): X raggiunge gli obiettivi;
allena(X): X si allena

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 in Prolog il predicato **`newList(L1,L)`** che data una lista **`L1`** di liste composte da due elementi interi (della forma **`[EL1, EL2]`**) oppure vuote, restituisca una nuova lista **`L`**, composta confrontando gli elementi **`EL1`** e **`EL2`** e inserendo in **`L`** il valore **`EL1`** se **`EL1>= EL2`**, **`EL2`** altrimenti. Se un elemento della lista **`L1`** è la lista vuota non andrà inserito in **`L`**. Se la lista **`L1`** è vuota anche la lista **`L`** sarà vuota.

Esempio:

```
?- newList([[3,7],[11,9], [], [12,3], [5,5]],L).
L = [7, 11, 12, 5]
?- newList([],L).
L = []
```

Esercizio 4 (6 punti)

Nel seguente programma Prolog, il predicato **listToSet**(L, S) trasforma una lista L in un insieme S eliminando gli elementi ripetuti di L, utilizzando anche il predicato **member**. Il programma è così definito:

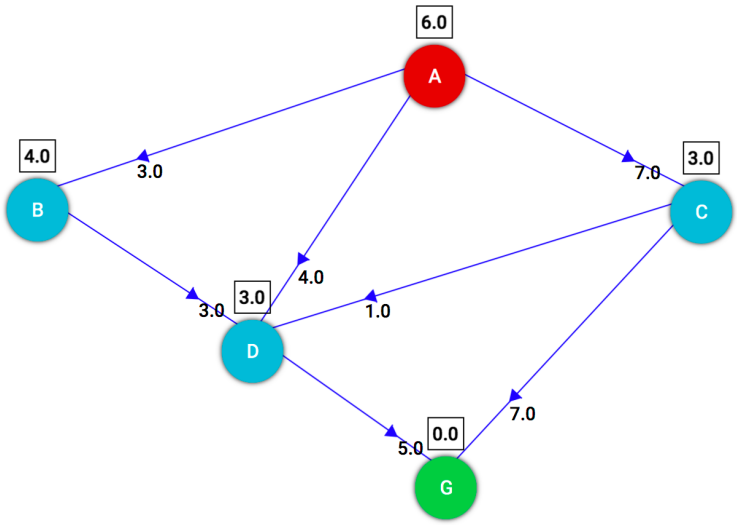
```
listToSet([],[]).
listToSet([H|T],Set) :-
    member(H,T), !,
    listToSet(T,Set).
listToSet([H|T], [H|Partial]) :-
    listToSet(T,Partial).

member(M, [M|_]) :- !.
member(M, [ _ |T]) :- member(M,T).
```

Si mostri l'*albero SLD* relativo al goal: **?- listToSet([1,1],S).**

Esercizio 5 (5 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;
- la condizione sulla stima euristica h(n) che garantisce l'ottimalità della ricerca su alberi e se è soddisfatta in questo caso.

Esercizio 6 (5 punti)

Dopo avere brevemente introdotto l'algoritmo di Forward Checking, se ne mostri l'esecuzione su questo esempio:

```
A::[1, 2, 3, 4]
B::[1, 2, 3, 4, 5, 6]
C::[1, 2, 3, 4, 5, 6]
A=3*B+1
C<=B+2
A<=10-C
```

considerando la prossima variabile da istanziare secondo l'euristica Minimum Remaining Values (MRV); a parità di MRV, si scelga la variabile che viene prima in ordine alfabetico. Nella scelta dei valori, si prediliga sempre di assegnare il valore minore.

Si mostri la riduzione dei domini delle variabili future dopo ogni labeling, fino alla prima soluzione.

15 Giugno 2023 - Soluzioni

Esercizio 1

Traduzione in logica
∀X (ammira(giorgia,X) → affidabile(X))
∀X (sportivo(X) and ¬ impegna(X) → ¬ obiettivi(X))
∃X sportivo(X) and ¬ allena(X)
∀X (sportivo(X) and ¬ allena(X) → ¬ impegna(X))
∀X (¬ obiettivi(X) → ¬ affidabile(X))

Query: ∃X ¬ ammira(giorgia,X).

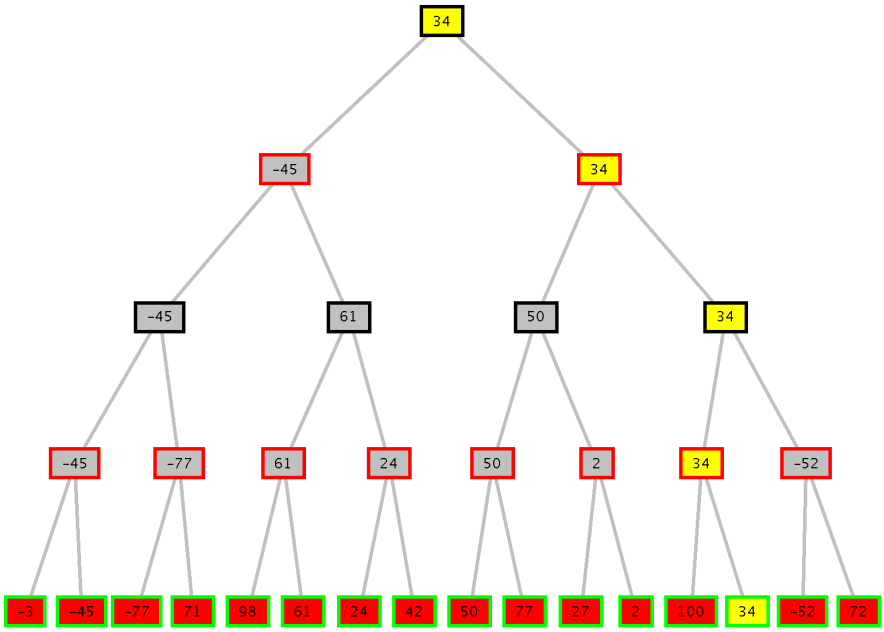
Trasformazione in clausole:
1. ¬ammira(giorgia,X) or affidabile(X)
2. ¬sportivo(X) or impegna(X) or ¬ obiettivi(X)
3. a) sportivo(c1). Skolem. b) ¬ allena(c1) Skolem
4. ¬sportivo(X) or allena(X) or ¬ impegna(X)
5. obiettivi(X) or ¬ affidabile(X)

GNeg: ammira(giorgia,X)

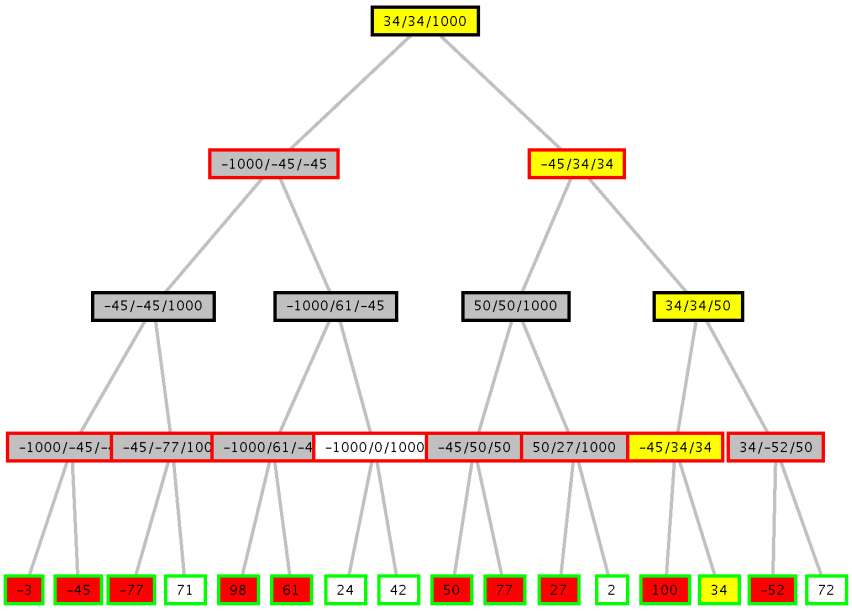
Risoluzione:
6.: 2+3a: impegna(c1) or ¬ obiettivi(c1)
7.: 3a+4: allena(c1) or ¬ impegna(c1)
8.: 7+3b: ¬impegna(c1)
9.: 8+6: ¬ obiettivi(c1)
10.: 9+5: ¬ affidabile(c1).
11.: 10+1: ¬ammira(giorgia, c1)
12.: 11+GNeg: [] contraddizione

Oppure risoluzione:
6.: GNeg+1: affidabile(X)
7.: 6+5: obiettivi(X)
8.: 7+2: ¬sportivo(X) or impegna(X)
9.: 8+4: ¬sportivo(X) or allena(X)
10.: 9+3a: allena(c1).
11.: 10+3b: [] contraddizione

Esercizio 2
Min-Max: in giallo la strada selezionata.



Alfa-Beta: In rosso i nodi espansi, in giallo la strada trovata, i nodi in bianco non sono esplorati per effetto dei tagli alfa-beta.

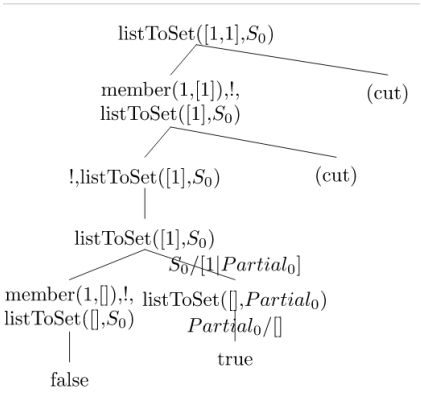


Esercizio 3

```
newList([],[]).
newList([[]|T],T1):-newList(T,T1).
newList([[X,Y]|T],[X|T1]):- X>=Y,!, newList(T,T1).
newList([[_,Y]|T],[Y|T1]):- newList(T,T1).
```

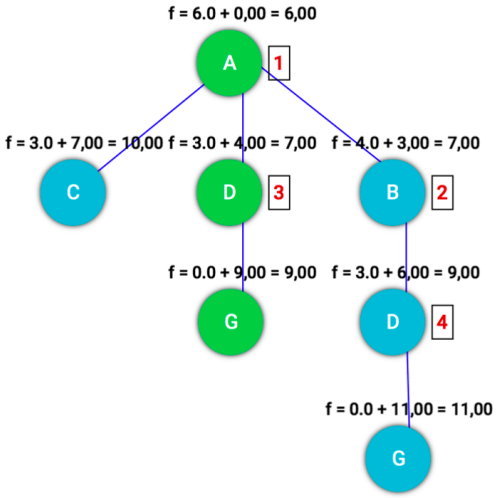
Esercizio 4

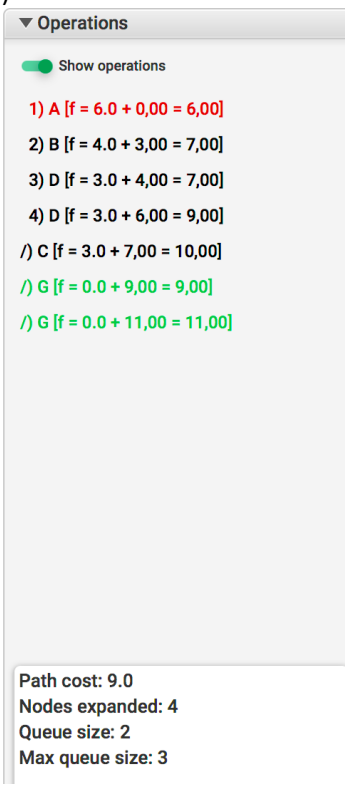
Goal: ?-listToSet([1,1],S).



Esercizio 5

Con A* i nodi espansi sono ABDDG e la soluzione ha costo 9 (ottimale):





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 FC.

	A	**B**	**C**
	[1..4]	**[1..6]**	**[1..6]**
Labeling	**A=1**		
FC	A=1	**Fail**	
Backtracking & Labeling	**A=2**		
FC		**Fail**	
Backtracking & Labeling	**A=3**		
FC		**Fail**	
Backtracking & Labeling	**A=4**		
FC	A=4	[1]	
Labeling	A=4	**B=1**	

FC	A=4	B=1	**[1...3]**
Labeling	A=4	B=1	**C=1**
Soluzione	**A=4**	**B=1**	**C=1**