FONDAMENTI DI INTELLIGENZA ARTIFICIALE

11 Luglio 2024 - Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)

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

1. *Giorgio raccoglie tutti i frutti maturi. (per qualunque Y che abbia un frutto X e X sia maturo, allora Giorgio lo raccoglie).*
2. *Tutti i ciliegi hanno almeno un frutto.*
3. *I frutti dei ciliegi rigogliosi sono maturi.*
4. *Esiste almeno un ciliegio rigoglioso.*

Si dimostri poi, tramite il principio di risoluzione, che Giorgio raccoglie qualcosa.

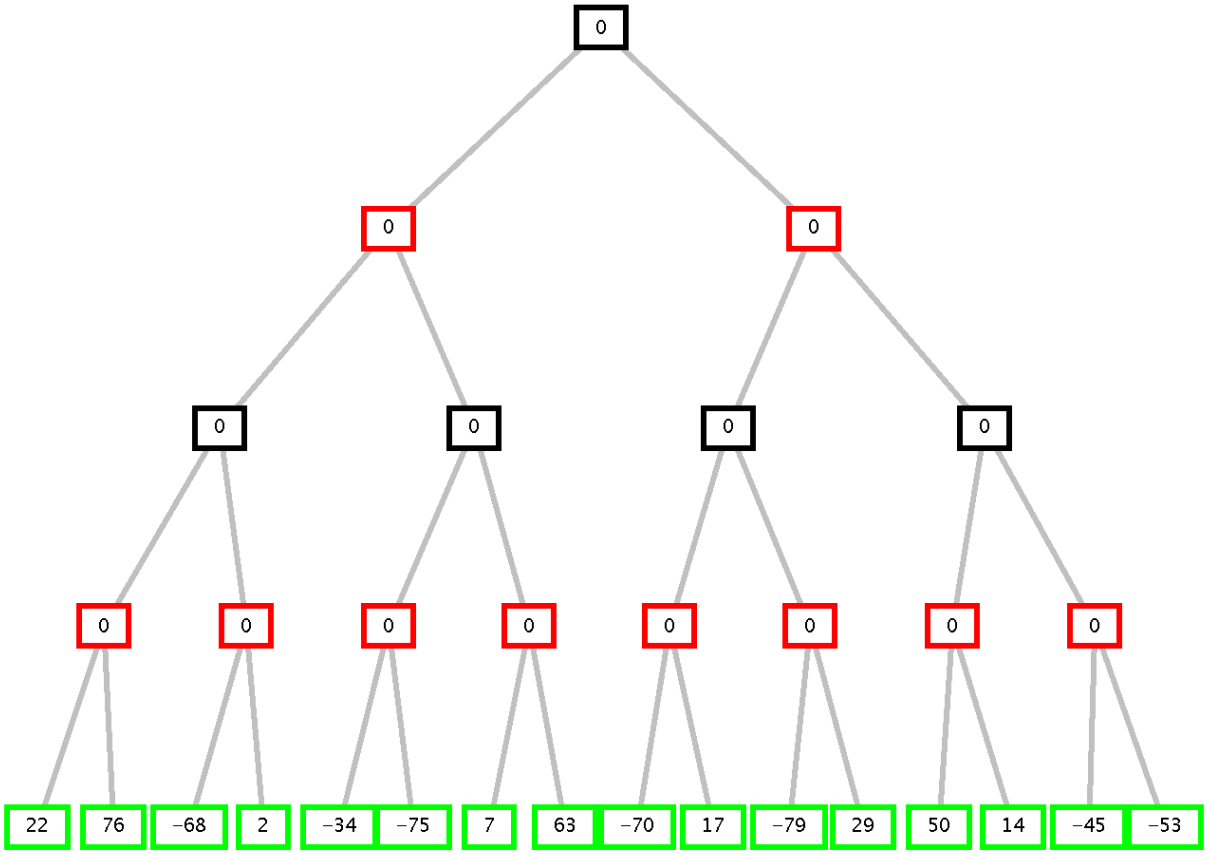
Si usino i seguenti predicati:

raccoglie(X,Y) "X raccoglie Y", **ciliegio(X)**, **rigoglioso(X), maturo(X), fruttaHa(Y,X)** "X ha il frutto Y."

Il nome "Giorgio" sia considerato la costante **giorgio.**

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 **delunicoLista(L1,L2)**, che data una lista **L1** di numeri interi, inserisca nella lista **L2** solo gli elementi di L1 che sono seguiti (nel seguito, ovvero nel resto della sequenza) da almeno un altro elemento uguale**.** Se **L1** è la lista vuota verrà restituita in uscita la lista vuota.

Per verificare se un elemento appartiene a una sequenza (resto) si usi il predicato **member/2**, definendolo e riportandolo nella soluzione.

Esempio:

```
?- delunicoLista([6,2,3,3,5,2,3,1,4] ,X).
X = [2, 3, 3]
```

Esercizio 4 (5 punti)

Dato il seguente predicato Prolog **maggioriLista(N,L1,L2)**, che dato il numero intero **N** e una lista **L1** di numeri interi, restituisce la lista **L2** contenente solo gli elementi della lista **L1** maggiori o uguali di **N**, e se **L1** è la lista vuota restituisce in uscita la lista vuota:

maggioriLista (_,[],[]).
maggioriLista (K,[H|T],[H|U]):- H>=K, maggioriLista (K,T,U),!.
maggioriLista (K,[_|T],U):- maggioriLista (K,T,U).

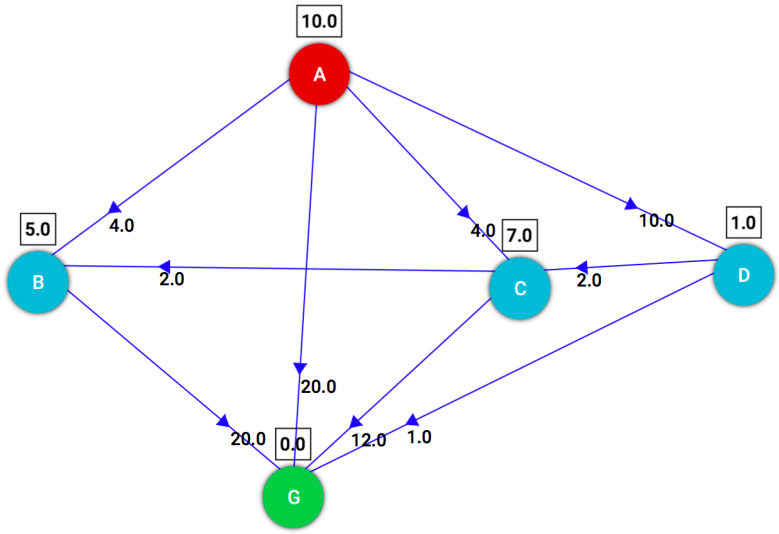
si mostri l'albero SLD generato dal goal:

?- maggioriLista(5,[4,6,3], X).

indicando eventuali rami di fallimento e quelli tagliati da cut, e la risposta calcolata per la variabile **X.**

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.

Si mostri poi cosa si ottiene se applica, invece, la ricerca **Greedy Best-first**, discutendo l'ottimalità o meno della soluzione individuata con questa seconda ricerca.

Esercizio 6 (5 punti)

Dopo avere brevemente introdotto l'algoritmo di Partial Look Ahead, si faccia vedere la sua applicazione sul problema in esame. Si considerino le variabili secondo il loro ordine alfabetico. Si ricorda che, dopo il labeling di ogni variabile, si applica sempre prima Forward checking e poi Partial Look Ahead.

A::[3, 4, 5, 6]
B::[1, 2, 3, 4]
C::[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
D::[2, 3, 4, 5]
A=B+1
C=B*2
A>D+1
C>D

11 Luglio 2024 - Soluzioni

Esercizio 1

Rappresentazione in FOL (e in clausole):

1. ∀X∀Y fruttaHa(X,Y) ∧ matura(X) ⟶ raccoglie(giorgio, X).
   ¬ fruttaHa(X,Y) ∨ ¬ matura(X) ∨ raccoglie(giorgio, X)
2. ∀M (ciliegio(M) ⟶ ∃B fruttaHa(B, M))
   ¬ciliegio(M) ∨ fruttaHa(f(M), M).
   f(M) funzione di Skolem
3. ∀M ∀B, ciliegio(M) ∧ rigoglioso(M) ∧ fruttaHa(B, M) ⟶ matura(B).
   ¬ciliegio(M) ∨ ¬rigoglioso(M) ∨ ¬fruttaHa(B, M) ∨ matura(B).
4. ∃M ciliegio(M) ∧ rigoglioso(M)
   4a. ciliegio(m) *m costante di Skolem*
   4b. rigoglioso(m)

Query: ∃X raccoglie(giorgio,X)

Query_neg.(Goal): ¬raccoglie(giorgio,X)

Risoluzione:

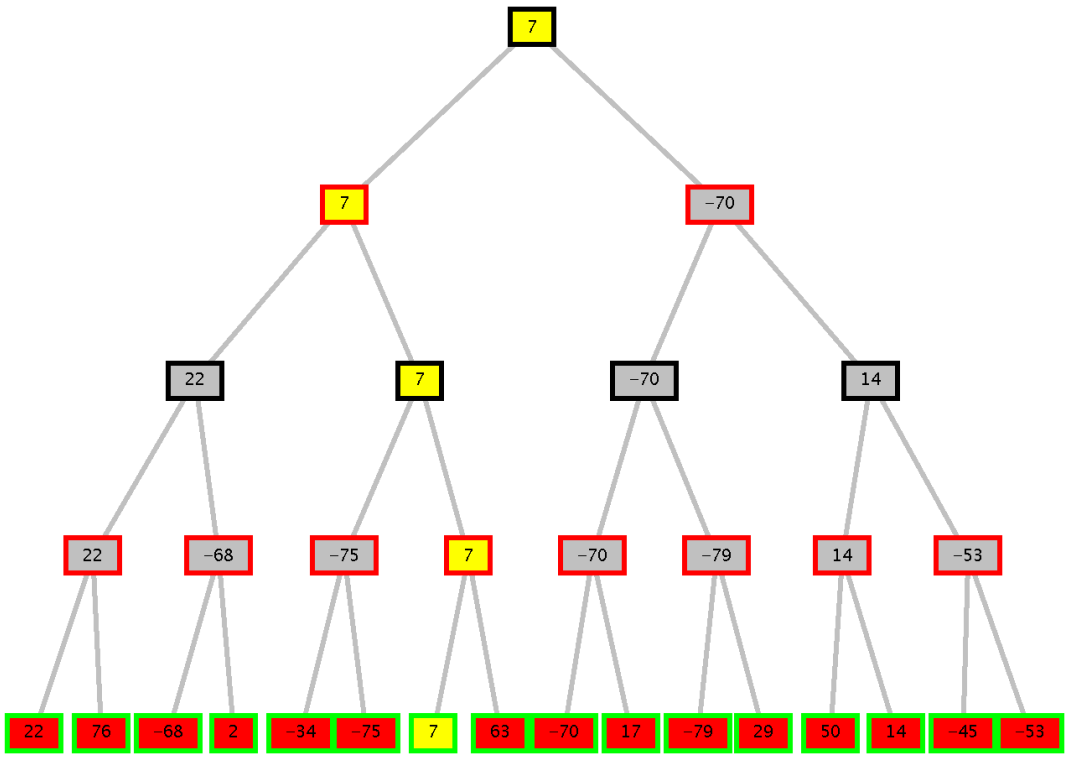
5: Gneg + 1: ¬ fruttaHa(X, Y) ∨ ¬ matura(X)
6:5+2: ¬ciliegio(M) ∨ ¬ matura(f(M)).
7: 6+3: ¬ciliegio(M) ∨ ¬rigoglioso(M) ∨ ¬fruttaHa(f(M), M)
8: 7+2: ¬ciliegio(M) ∨ ¬rigoglioso(M).
9: 8+4a: ¬rigoglioso(m)
10: 9+4b: Contraddizione!!

Oppure:

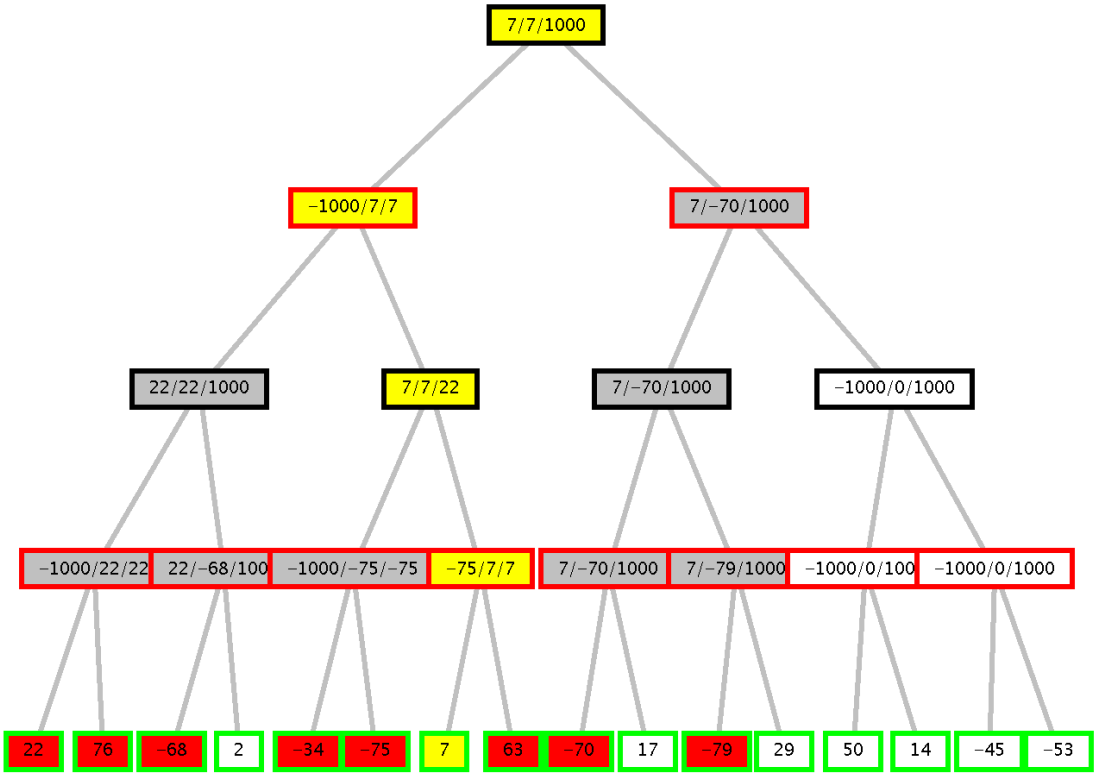
5: Gneg + 1: ¬ fruttaHa(X, Y) ∨ ¬ matura(X)
6: 5+3: ¬ciliegio(M) ∨ ¬rigoglioso(M) ∨ ¬fruttaHa(X, M)
7: 6+2: ¬ciliegio(M) ∨ ¬rigoglioso(M)
8: 7+4a: ¬rigoglioso(m)
9: 8+4b: Contraddizione.

Esercizio 2

Min max:



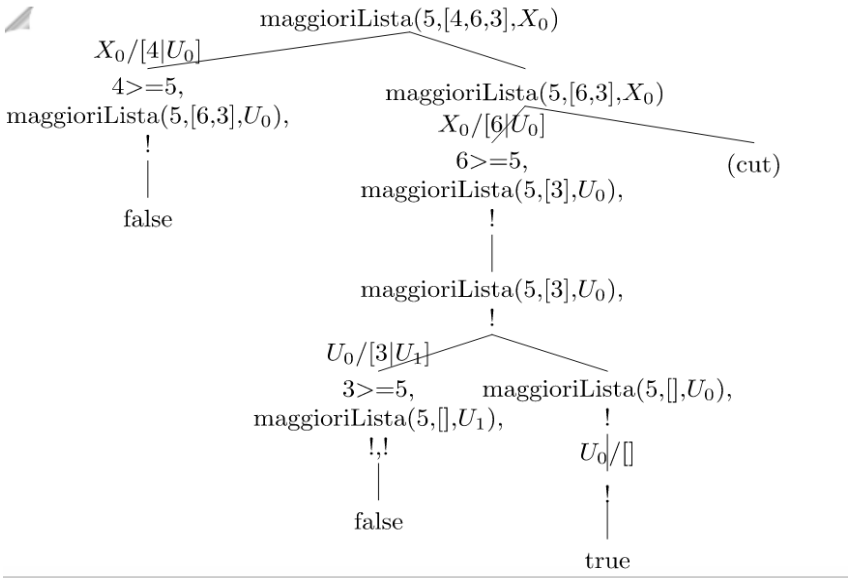
Alfa Beta:



Esercizio 3

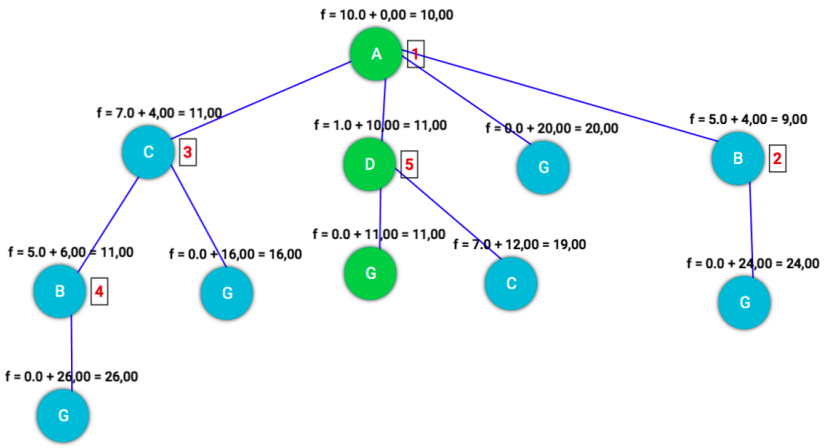
```
member(N,[N|_]):-!.
member(N,[_|T]):-member(N,T).

delunicoLista([],[]).
delunicoLista([H|T],[H|Y]):-
   member(H,T),
   !,
   delunicoLista(T,Y).
delunicoLista([_|T],Y):-
   delunicoLista(T,Y).
```

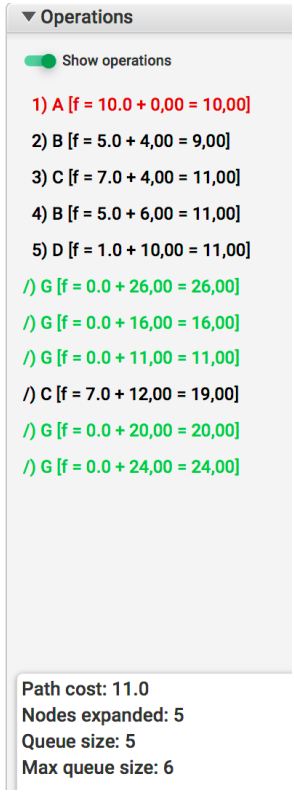
Esercizio 4



Risposta calcolata: X=[6]

Esercizio 5
Con A*:

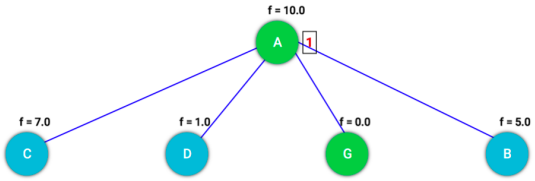


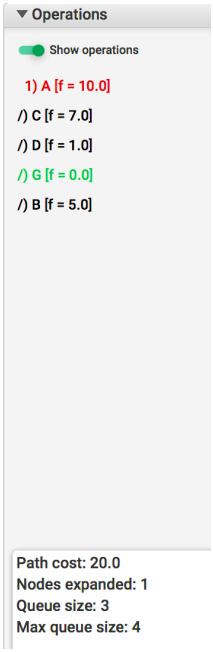


- I nodi espansi nell'ordine di espansione sono: ABCBDG
- I nodi sulla strada della soluzione sono ADG e il costo della soluzione è 11;

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.

Gready Best-first:





I nodi sulla soluzione sono AG, la soluzione trovata ha costo 20 (non è ottima).

Esercizio 6

Vedi slide del corso per spiegare Partial Look Ahead.

A=B+1
C=B*2
A>D+1
C>D

	A	**B**	**C**	**D**
	[3..6]	[1..4]	[1..10]	[2..5]
Labeling	**A=3**			
FC		**[2]**		Fail*
Backtracking				
Labeling	**A=4**			
FC		**[3]**		**[2]**
PLA			**[3..10]**	
Labeling		**B=3**	[6]	[2]
FC				
PLA				
Labeling			**C=6**	
FC				
PLA				
Labeling				**D=2**
Soluzione	**A=4**	**B=3**	**C=6**	**D=2**

*: Siccome si verifica un fallimento già nella fase di FC, non si arriva alla eventuale fase di PLA.