FONDAMENTI DI INTELLIGENZA ARTIFICIALE

13 Febbraio 2020 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (6 punti)

Si rappresentino in logica dei predicati del I ordine, le seguenti affermazioni:

- A tutte le maestre piacciono gli alunni
- Tutti gli alunni sono bambini
- Alle maestre non piacciono i bambini maleducati
- Giovanna è una maestra

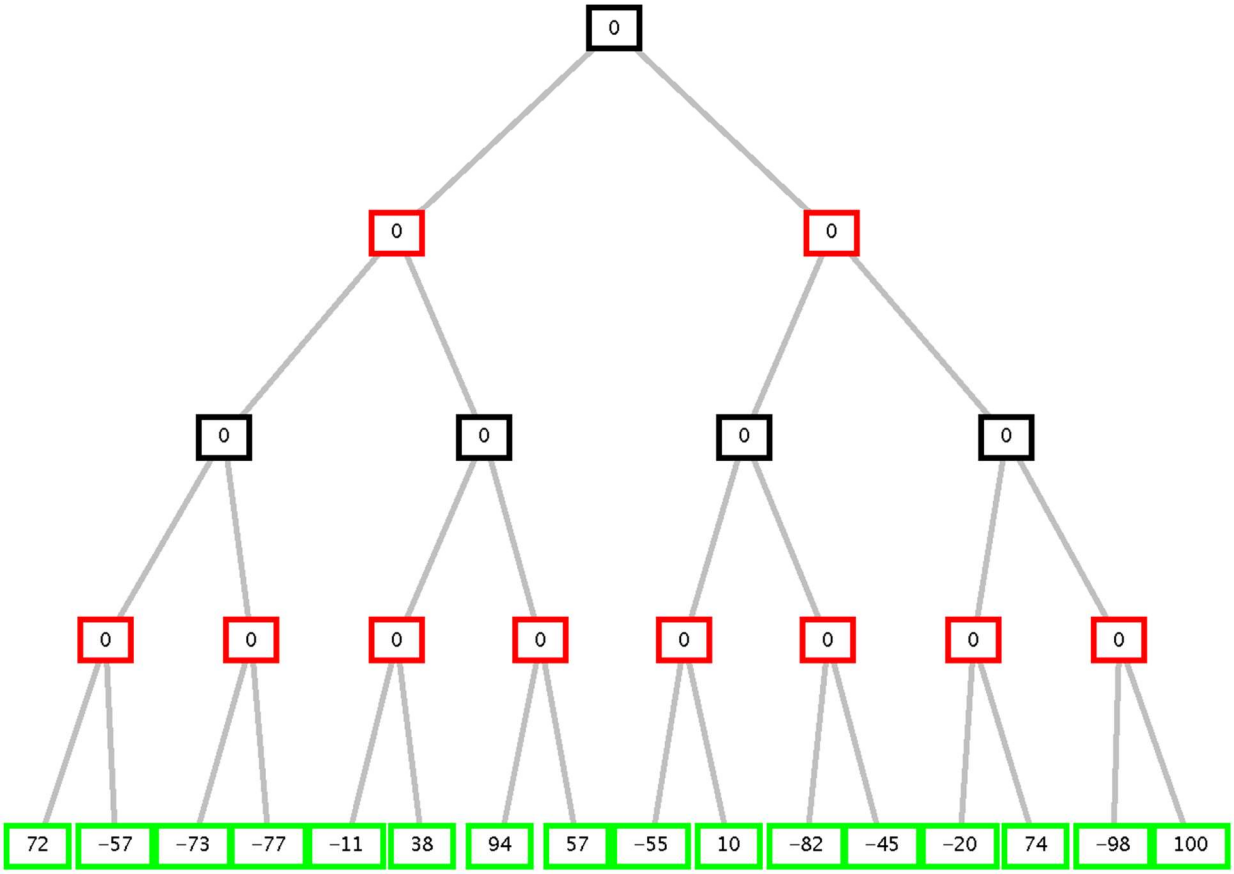
Si applichi la risoluzione alla teoria formata dalle formule ottenute sopra, poste in forma a clausole, per verificare che:

- Tutti gli alunni non sono maleducati.

Si utilizzino i predicati `alunno(X)`, `maestra(X)`, `maleducato(X)` , `piace(X,Y)` (a X piace Y), `bambino(X)` con l'ovvio significato.

Esercizio 2 (5 punti)

Si consideri il seguente albero di gioco in cui il primo giocatore è *MAX*. Si mostri come l'algoritmo *min-max* e l'algoritmo *alfa-beta* risolvono il problema e la mossa selezionata dal primo giocatore.



Esercizio 3 (6 punti)

Dato il seguente programma Prolog, `sumNeg(L,B,L1)` che, data la lista `L`, e il numero intero `B`, produce in uscita la lista `L1` in cui ad ogni elemento minore di 0 di `L` è stato sommato il numero `B`:

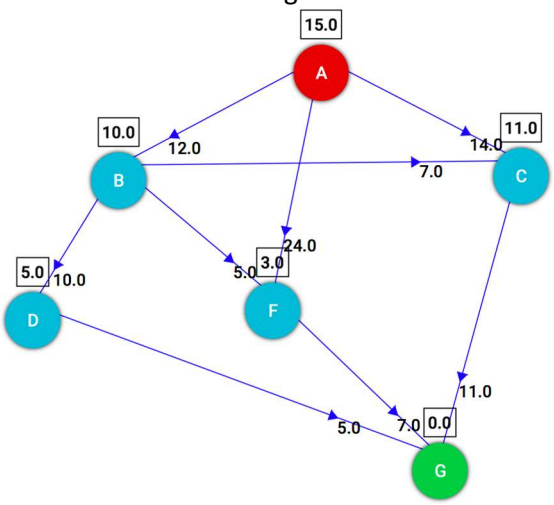
```
sumNeg([],_,[]).
sumNeg([H|T],B,[H1|T1]):- H<0, !, H1 is H+B,
                           sumNeg(T,B,T1).
sumNeg([H|T],B,[H|T1]):- sumNeg(T,B,T1).
```

disegnare l'albero SLD per il goal seguente (si indichino i tagli effettuati dal *cut* e non si espandano gli eventuali rami tagliati):

```
?- sumNeg([1,-10,3],5, L1).
```

Esercizio 4 (5 punti)
Si considerino due liste L1 e L2, formate a sua volta da liste di numeri interi. Scrivere un programma PROLOG appendList(L1,L2,L3) che, date le liste L1 e L2 produca in uscita la lista di liste L3, in cui ogni elemento è a sua volta una lista costruita appendendo un elemento (lista) di L2 al corrispondente elemento (lista) di L1. Se una delle due liste, L1 o L2, è vuota (o diventa vuota), la L3 risultante sarà semplicemente uguale alla lista (tra le due, L1 e L2, date in input) non vuota.
Ad esempio alla query:
?- appendList([[1,5],[1,3]], [[7,5], [1,3,9], [4,5], []], L3).
la risposta sarà:
yes 3=[[1,5,7,5], [1,3,1,3,9], [4,5], []]
Si mostri la realizzazione in Prolog di tutti i predicati utilizzati.

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:



a) Si applichi la ricerca **greedy best first** e si disegni l'albero di ricerca sviluppato indicando per ogni nodo *n* l'ordine di espansione. In caso di non-determinismo, si scelgano i nodi da espandere in base all'ordine alfabetico del loro nome. Si consideri come euristica h(*n*) quella indicata nel quadrato a fianco di ogni nodo in figura.
b) Qual è il costo di cammino trovato ed il numero di nodi espansi per arrivare al goal G a partire dal nodo iniziale A?
c) La soluzione è ottimale? (motivare la risposta)

Esercizio 6 (4 punti)
Si consideri il seguente **map coloring problem**, con 4 variabili X1, X2, X3, X4 in cui X1 e X3 possono essere colorate di giallo o verde, mentre X2 e X4 di rosso o verde. Si devono poi rispettare i seguenti vincoli di diversità:
X1 =/= X2
X1 =/= X3
X1 =/= X4
X2 =/= X3
X2 =/= X4
X3 =/= X4
Dopo aver dato la definizione di arc-consistenza e disegnato il constraint graph per il problema enunciato, si mostri se il problema CSP è arc-consistente o se, invece, è necessario renderlo arc-consistente cancellando alcuni valori dai domini delle variabili. Il problema in generale ammette soluzione?

13 Febbraio 2020 - Soluzioni

Esercizio 1

1.: ∀X ∀Y alunno(X) ∧ maestra(Y) ⟶ piace(X, Y).
2.: ∀X alunno(X) ⟶ bambino(X).
3.: ¬∃X (maestra(X) ∧ ∃Y (maleducato(Y) ∧ piace(X, Y)))
Oppure
3.: ∀X ∀Y maestra(X) ∧ bambino(Y) ∧ maleducato(Y)⟶ ¬ piace(X, Y).
4.: maestra(giovanna).
Goal:
∀X alunno(X) ⟶ ¬maleducato(X).
oppure, in modo equivalente: ¬∃X (alunno(X) ∧ maleducato(X))

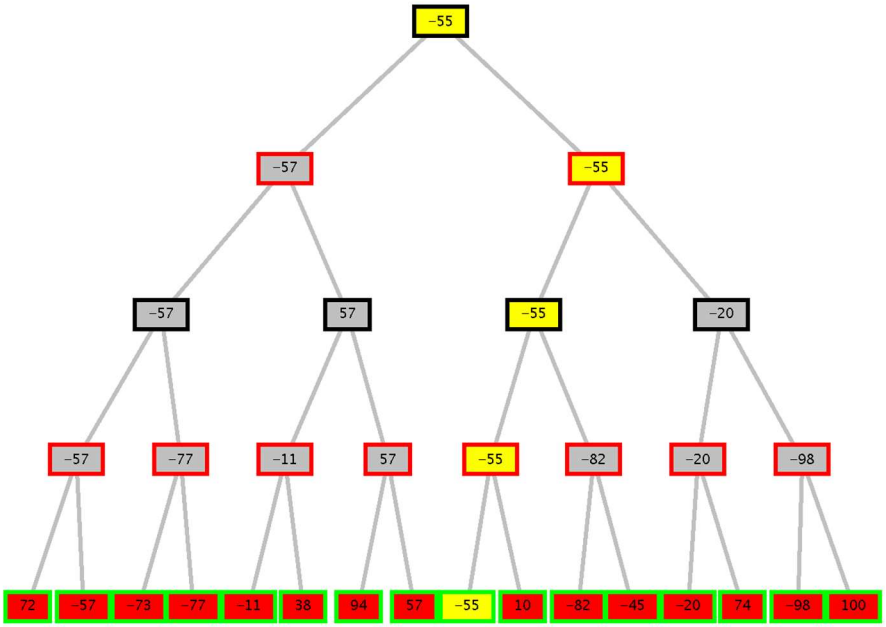
Goal negato: ¬∀X (alunno(X) ⟶ ¬maleducato(X))

Tradotti in clausole:
1.: ¬alunno(X) ∨¬maestra(Y) ∨ piace(X, Y).
2.: ¬alunno(X) ∨ bambino(X).
3.: ¬ maestra(X) ∨ ¬ maleducato(Y) ∨ ¬ bambino(Y) ∨ ¬piace(X, Y)
4.: maestra (giovanna)
GNeg1: alunno(c1) skolem
GNeg2: maleducato(c1)

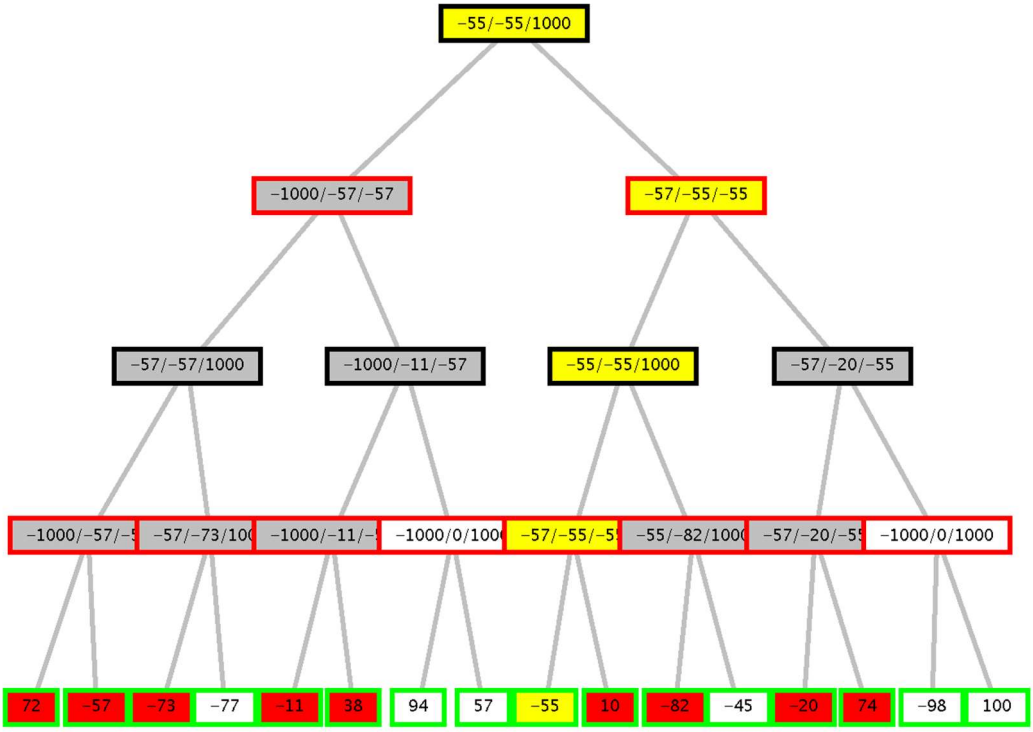
Refutazione:
5.: GNeg1+1: ¬maestra(X) ∨ piace(X,c1)
6.: 3+ GNeg2: ¬maestra(X) ∨ ¬piace(X, c1) ∨ ¬bambino(c1)
7.: 6 + 4: ¬piace(giovanna, c1) ∨ ¬ bambino(c1)
8.: 5 + 4: piace(giovanna, c1)
9.: 7 + 8: ¬ bambino(c1)
10.: 9 + 2: ¬ alunno(c1)
11: 10 + Gneg1: Contraddizione logica

Esercizio 2

Min-max:



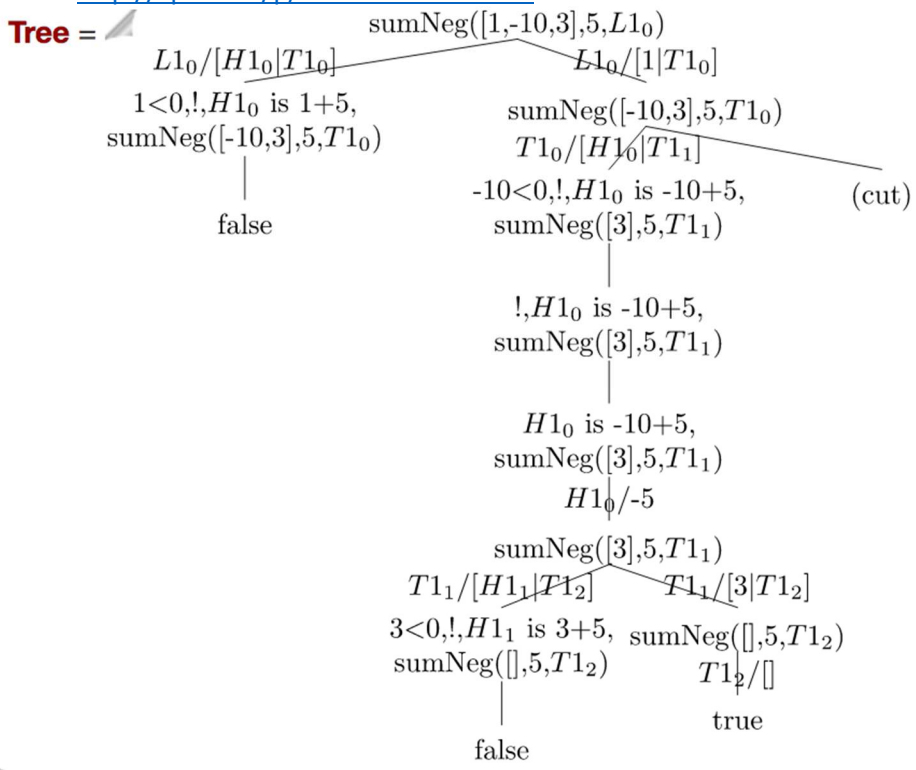
Alfa-beta:



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

Soluzione generata con http://cplint.eu/p/mGWdAOCw.swinb

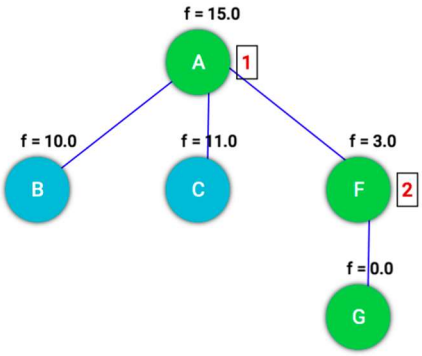


Esercizio 4

```
appendList([],L2,L2).
appendList(L1,[],L1).
appendList([H1|T1],[H2|T2], [H3|T3]):-
      append(H1,H2,H3), appendList(T1,T2,T3).

append([], X, X).
append([H|T], X, [H|S]) :- append(T, X, S).
```

Esercizio 5



Il costo di cammino trovato è 31 e il numero di nodi espansi per arrivare al goal G a partire dal nodo iniziale A è 2 (AF) non considerando il goal test sul goal G. Non viene trovata la soluzione ottimale (24) in quanto la ricerca best-first non la garantisce considerando ad ogni passo di espansione per la selezione del prossimo nodo solo la stima della sua distanza dal goal ma non il costo del cammino percorso per arrivarci.

Esercizio 6

Per la definizione di arc-consistenza (si vedano le slide del corso).

Si può mostrare applicando l'argoritmo di arc-consistency, che il constraint-graph risultante (da disegnare nella soluzione) è arc-consistente così come è, ma comunque non ammette soluzione.