**FONDAMENTI DI INTELLIGENZA ARTIFICIALE**

1 Febbraio 2023– Tempo a disposizione: 2 h – Risultato: 32/32 punti

**Esercizio 1 (6 punti)**

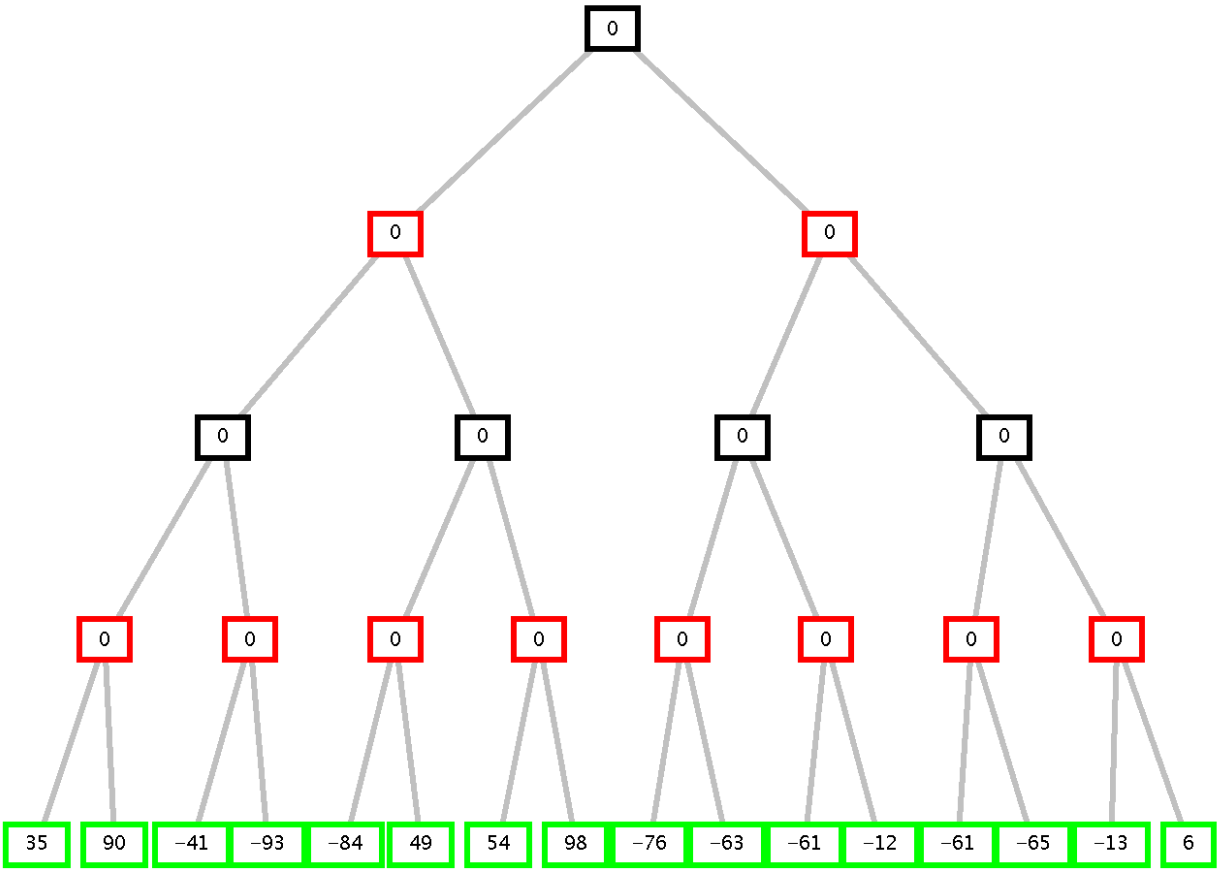
Si formalizzino le seguenti frasi in logica dei predicati del I ordine:

- I giorni sono feriali  o (xor) festivi
- I giorni feriali sono di lavoro
- I giorni festivi sono di vacanza
- Lunedì è un  giorno
- Lunedì non è un giorno festivo
- Lunedì è il mio compleanno
- Esiste un giorno festivo

Le si trasformi in clausole e si usi la risoluzione per dimostrare che esiste il mio compleanno in un giorno che è di lavoro. A tal scopo si usino i predicati unari giorno, feriale, festivo, lavoro, vacanza, compleanno, e la costante lunedì.

**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 (6 punti)**
Si scriva in Prolog il predicato **controlla**(L1,L2,L3) che date due liste L1 e L2 di interi (di lunghezza identica) restituisca una nuova lista L3, della stessa lunghezza e con elementi 0 e 1, composta confrontando gli elementi EL1 e EL2 in medesima posizione in L1 e L2 rispettivamente e inserendo in L3 il valore 1 se El1>= El2, 0 altrimenti.
Esempio:

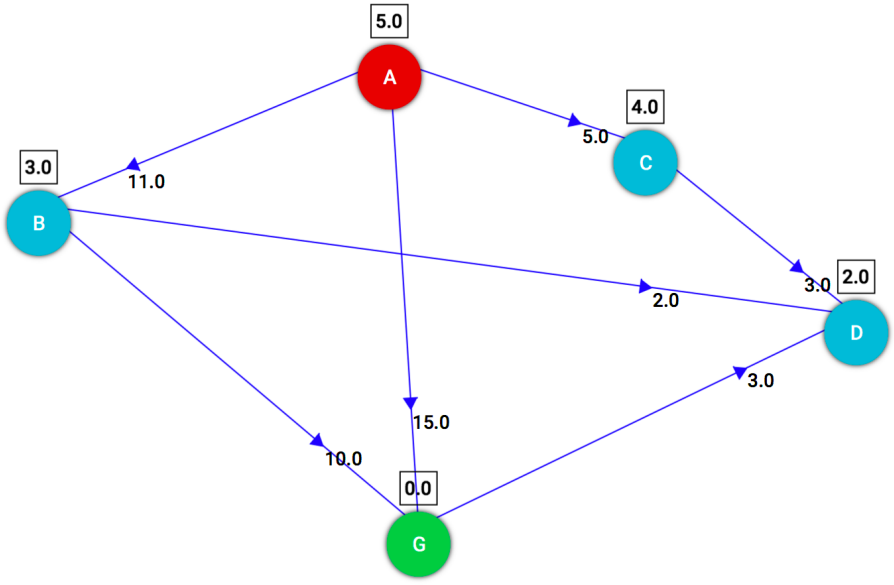
```
?-controlla([2,5,9],[4,2,7],L).
L = [0, 1, 1]
?-controlla([2,5],[4,2],L).
L = [0, 1]
?-controlla([],[],L).
L = []
```

**Esercizio 4 (6 punti)**
Nel seguente programma Prolog il predicato **member(Element,L)** è vero quando **Element** appartiene alla lista **L**:

```
member(X ,[X|_T]).
member(X ,[_H|T]):-
        member(X,T).
```

Si mostri l'*albero SLD* relativo al goal: **?-member(X,[4]), \+member(X,[1,3]).**
Nota: \+ è il not.

**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) **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;
- se è garantita o meno l'ottimalità.

**Esercizio 6 (4 punti)**
Dopo avere brevemente introdotto l'algoritmo di Forward Checking, se ne mostri l'esecuzione su questo esempio:
A,B,C ::[1,2,3,4]
B<A
C<B
A=C+2
Considerando le variabili secondo l'ordine alfabetico (nel passo di labeling) e mostrando la riduzione dei domini delle variabili future dopo ogni labeling, fino alla prima soluzione.

**1 Febbraio 2023 - Soluzioni**

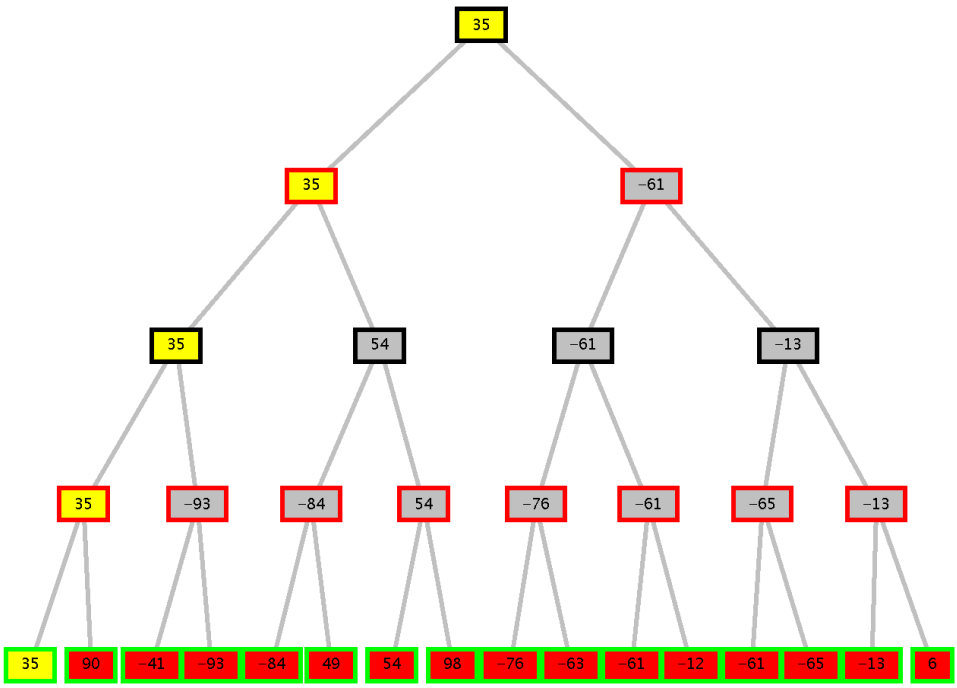
**Esercizio 1**
Traduzione in predicati in **logica del primo ordine**:
∀X giorno (X) -> (feriale(X) ex-or festivo(X))
∀X giorno (X) and feriale(X) -> lavoro(X)
∀X giorno(X) and festivo(X) -> vacanza(X)
giorno(lunedì)
not festivo(lunedì)
compleanno(lunedì)
∃X giorno(X) and festivo(X)

Query: ∃X giorno(X) and compleanno(X) and lavoro(X)
Query negata (goal): not ∃X giorno(X) and compleanno(X) and lavoro(X).

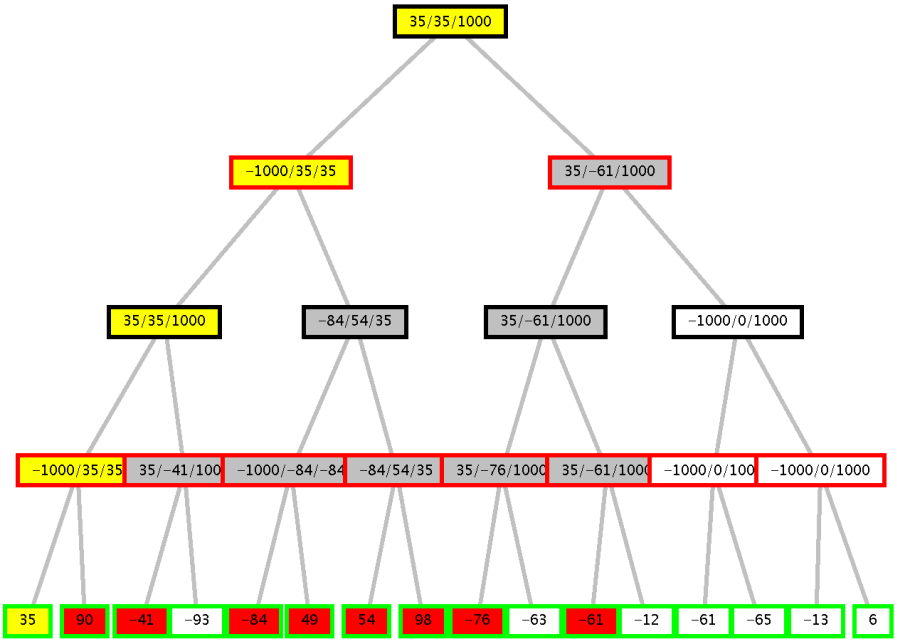
**Clausole:**
1. not giorno (X) or feriale(X) or festivo(X)
2. not giorno(X) or not feriale(X) or not festivo(X)
3. not giorno(X) or not feriale(X) or lavoro(X)
4. not giorno(X) or not festivo(X) or vacanza(X)
5. giorno(lunedì)
6. not festivo(lunedì)
7. compleanno(lunedì)
8. festivo(c1) *// Costante di Skolem c1*
9. *giorno(c1)*
10. not giorno(X) or not compleanno(X) or not lavoro(X)

**Risoluzione:**
11: 10+7 not giorno(lunedì) or not lavoro(lunedì)
12: 11 + 3: not giorno(lunedì) or not feriale( lunedì)
13: 12+ 1: not giorno (lunedì) or festivo(lunedì)
14: 13 + 6: not giorno(lunedì)
15: 14 + 5: contraddizione!!

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

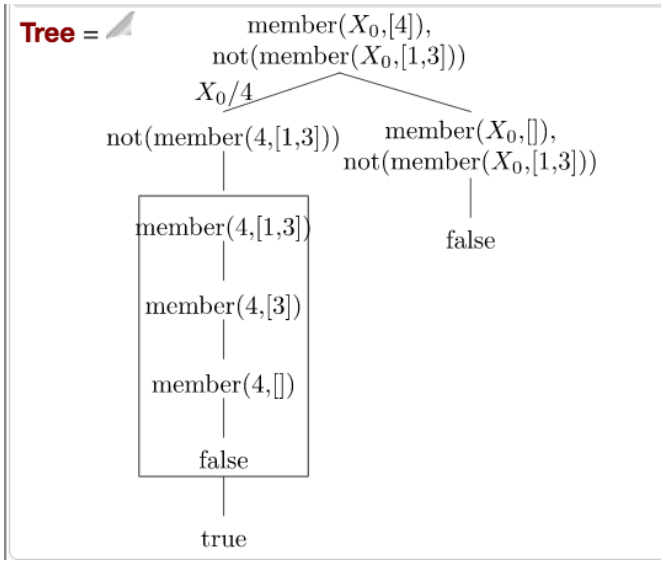


Alfa-Beta: In rosso i nodi espansi, in giallo la strada trovata, i nodi in bianco non sono esplorati per effetto dei tagli alfa-beta.

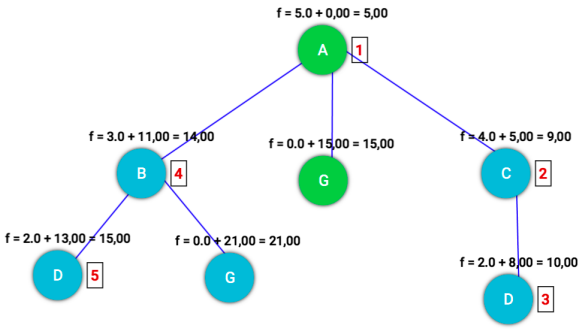


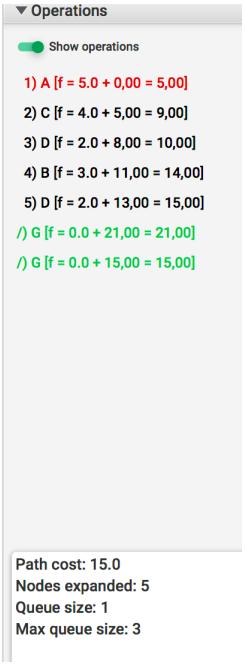

**Esercizio 3**
controlla([], [], []) :- !.
controlla([A|T], [B|L], [1|M]) :- A>= B, !, controlla(T, L, M).
controlla([_|T], [_|L], [0|M]) :- controlla(T, L, M).

**Esercizio 4**
**Goal:** member(X,[4]), \+member(X,[1,3]).




**Esercizio 5**
Con A* i nodi espansi sono ACDBDG e la soluzione ha costo 15 (ottimale):




**Esercizio 6**

Vedi slide del corso per spiegare FC.

A,B,C ::[1,2,3,4]

B<A

C<B

A=C+2

| | **A** | **B** | **C** |
|---|---|---|---|
| | [1..4] | [1..4] | [1..4] |
| Labeling | A=1 | [1...4] | [1...4] |
| FC e Backtracking | A=1 | **Fail** | [1...4] |
| Labeling | A=2 | [1..4] | [1...4] |
| FC e Backtracking | A=2 | [1] | **Fail** |
| Labeling | A=3 | [1...4] | [1...4] |
| FC | A=3 | [1,2] | [1] |
| Labeling | A=3 | B=1 | [1] |
| FC e Backtracking | A=3 | B=1 | **Fail** |
| Labeling | A=3 | B=2 | [1] |
| FC | A=3 | B=2 | [1] |
| Labeling | **A=3** | **B=2** | **C=1** |