

**Esercizio 1 (6 punti)**

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

*Tutti coloro che sono soli sono tristi*

*Tutti sono tristi o contenti, ma non entrambe le cose*

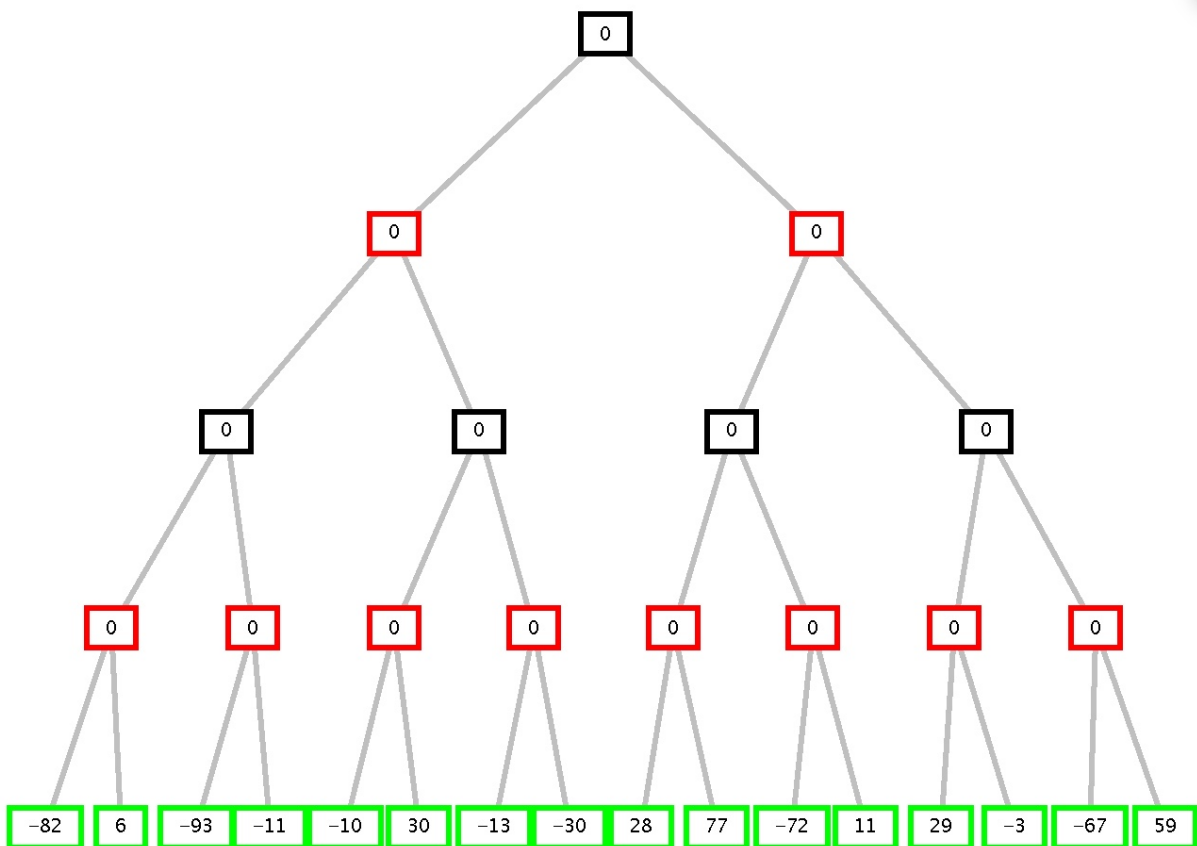
*Esiste qualcuno non solo*

*Beatrice è sola*

utilizzando i predicati: *solo*(X), *triste*(X), *contento*(X). Si trasformino poi le formule in clausole, e si dimostri, applicando il principio di risoluzione, che: *Esiste qualcuno non contento*.

**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 consideri il seguente CSP che lega le variabili A, B, C, D:

- A::[4, 5, 6]                      A>B-3
- B::[1, 2, 3, 4, 5]              C>B-5
- C::[1, 2, 3, 4, 5]              A=D+4
- D::[1, 2, 3, 4, 5]              C>D+3

Si applichi, durante la ricerca fino alla prima soluzione, il Forward Checking dopo ogni passo di labeling, considerando, nella scelta della prossima variabile da istanziare l’euristica **Minimum Remaining Value** (poi, a parità di cardinalità di dominio, scegliere in base all’ordine alfabetico dei nomi delle variabili). Nel labeling, per il valore da assegnare alla variabile, si considerino i valori di dominio in ordine crescente, partendo dal più piccolo.

#### Esercizio 4 (5 punti)

Si scriva in Prolog il predicato `inserisciOrd(N,L1,L2)`, che dato il numero intero **N** e una lista ordinata **L1** (in modo crescente, eventualmente vuota) di numeri interi, restituisca in **L2** la lista ordinata ottenuta inserendo **N** in **L1**. Esempio:

?- `inserisciOrd(3, [1,2,5], X)`.

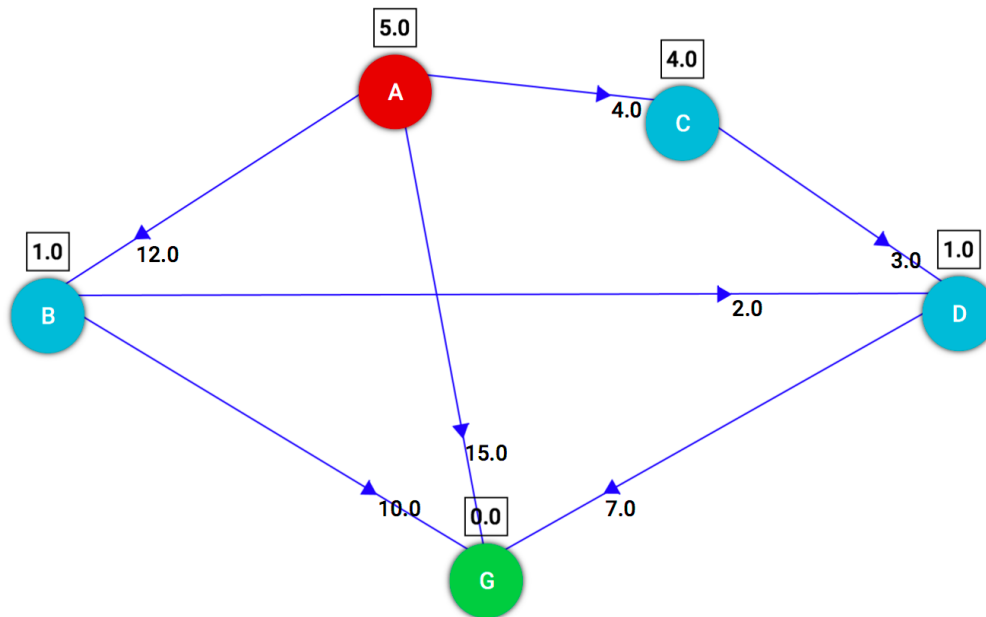
**X** = [1, 2, 3, 5]

?- `inserisciOrd(3, [], X)`.

**X** = [3]

#### Esercizio 5 (7 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 e indicando:

- 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 e perchè.

Cosa cambierebbe con ricerca **Greedy-Best-First**?

#### Esercizio 6 (4 punti)

Si consideri il programma Prolog seguente:

`p:-q.`

`p:-r.`

`r:-t.`

`r:-s.`

`s.`

e il goal

?- `not p.`

Si spieghi brevemente come si costruisce l'albero SLDNF e si disegni l'albero SLDNF per il goal e il programma dati.

**Esercizio 1**

Traduzione in logica

1.  $\forall Y \text{ solo}(Y) \Rightarrow \text{triste}(Y)$ .
  2.  $\forall X \text{ triste}(X) \text{ OR-EX } \text{contento}(X)$ .
  3.  $\exists X \neg \text{solo}(X)$ .
  4.  $\text{solo}(\text{beatrice})$ .
- Goal:  $\exists X \neg \text{contento}(X)$ .

Trasformazione in clausole

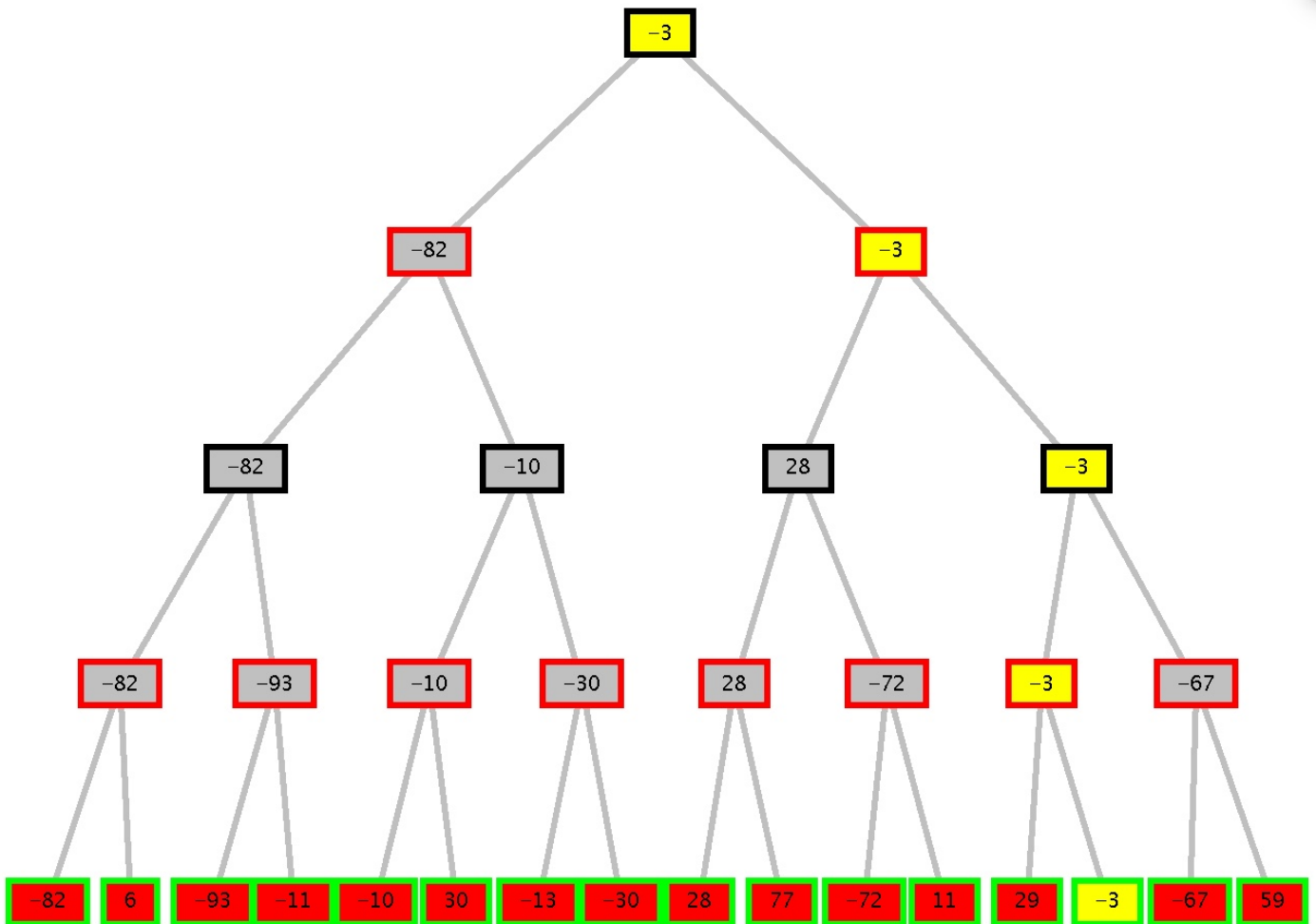
- 1.:  $\text{triste}(Y) \vee \neg \text{solo}(Y)$ .
  - 2a.:  $\text{triste}(X) \vee \text{contento}(X)$ .
  - 2b.:  $\neg \text{triste}(X) \vee \neg \text{contento}(X)$ .
  - 3.:  $\neg \text{solo}(c1)$ . (costante di Skolem)
  - 4.:  $\text{solo}(\text{beatrice})$ .
- GNeg.:  $\text{contento}(X)$ .

Risoluzione

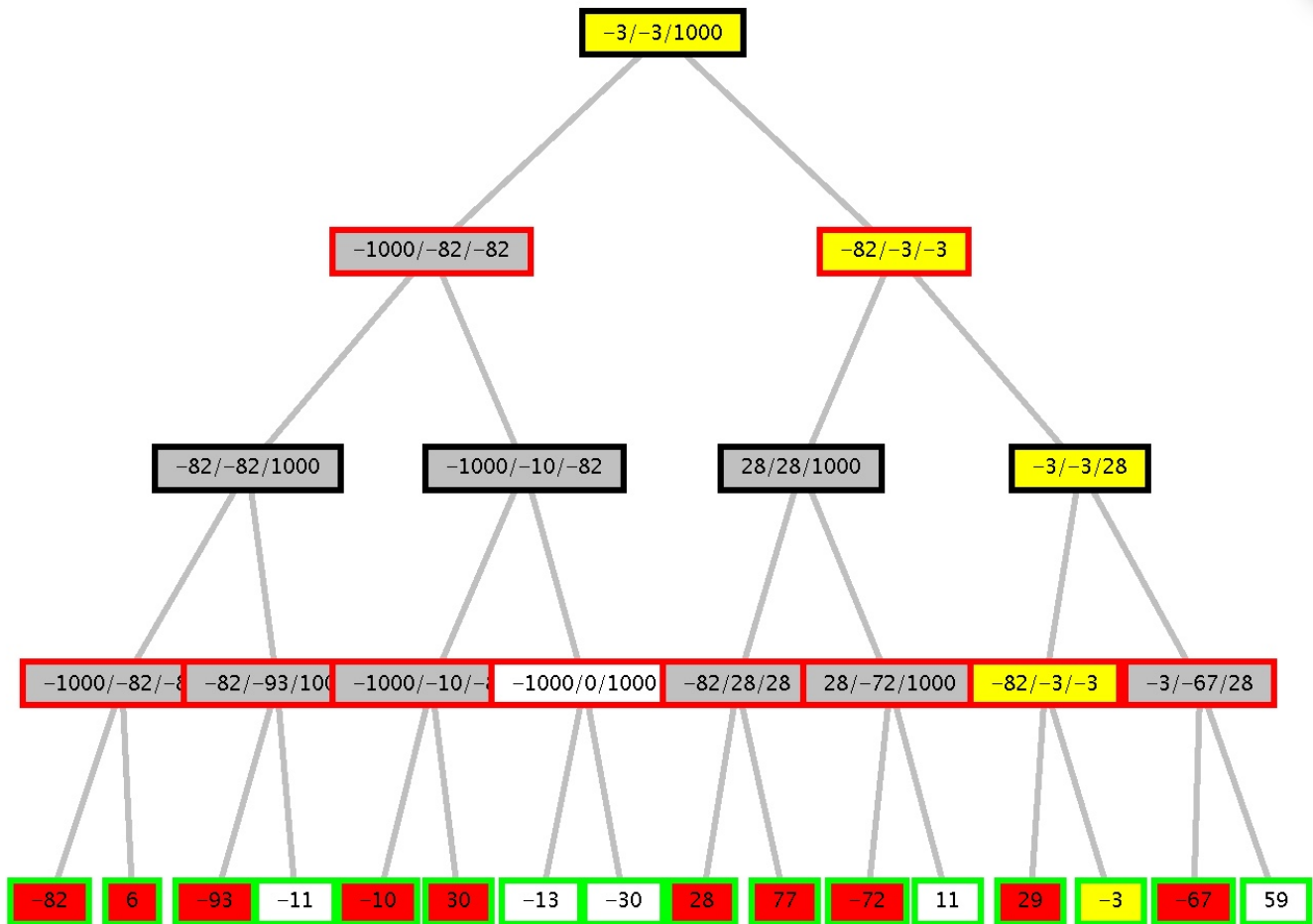
- 5.: GNeg. + 2b:  $\neg \text{triste}(X)$ .
- 6.: 5 + 1:  $\neg \text{solo}(X)$ .
- 7.: 6 + 4: **contraddizione!!**

**Esercizio 2**

Min-max: strada in giallo, valore nodo radice -3, ramo a destra.



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.



Archi tagliati a10, a18, a26, a 30 (4 tagli). Scelta per il ramo a2, valore propagato -3.

### Esercizio 3

Con euristica MRV:

	A	B	C	D
Labeling	A=4	[1...5]	[1...5]	[1...5]
FC e Backtracking	A=4	[1...5]	[1...5]	Fail
Labeling	A=5	[1...5]	[1...5]	[1...5]
FC	A=5	[1...5]	[1...5]	[1]
Labeling	A=5	[1...5]	[1...5]	D=1
FC	A=5	[1...5]	[5]	D=1
Labeling	A=5	[1...5]	C=5	D=1
FC	A=5	[1...5]	C=5	D=1
Labeling	A=5	B=1	C=5	D=1
<b>Soluzione</b>	<b>A=5</b>	<b>B=1</b>	<b>C=5</b>	<b>D=1</b>

### Esercizio 4

```

inserisciOrd(N, [], [N]) :- !.
inserisciOrd(N, [H|T], [N,H|T]) :- N =< H, !.
inserisciOrd(N, [H|T], [H|Z]) :- inserisciOrd(N,T,Z).

```

OPPURE:

```

inserisciOrd(N, [], [N]) :- !.
inserisciOrd(N, [H|T], [H|T1]) :- N > H, !, inserisciOrd(N,T,T1).
inserisciOrd(N,L, [N|L]).

```

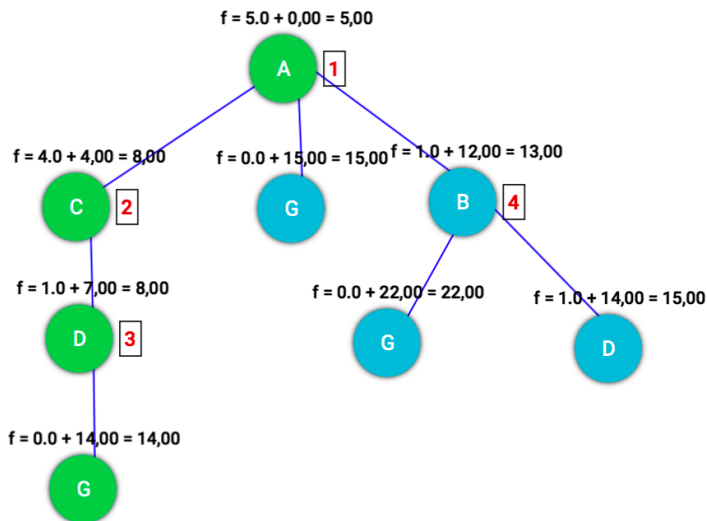
OPPURE:

```

inserisciOrd(N,[H|T],[H|T1]) :- N > H, !, inserisciOrd(N,T,T1).
inserisciOrd(N,L,[N|L]).
    
```

**Esercizio 5**

Con A\*, i nodi espansi sono ACDBG, la soluzione trovata ACDG è ottimale perché l'euristica è ammissibile e il costo della soluzione trovata è 14:



▼ Operations

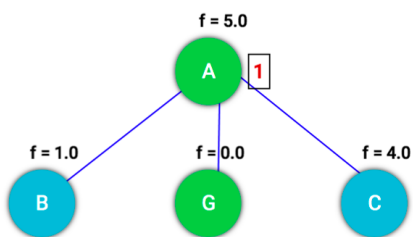
Show operations

- 1) A [f = 5.0 + 0,00 = 5,00]
- 2) C [f = 4.0 + 4,00 = 8,00]
- 3) D [f = 1.0 + 7,00 = 8,00]
- 4) B [f = 1.0 + 12,00 = 13,00]

- /) G [f = 0.0 + 14,00 = 14,00]
- /) G [f = 0.0 + 15,00 = 15,00]
- /) G [f = 0.0 + 22,00 = 22,00]
- /) D [f = 1.0 + 14,00 = 15,00]

Path cost: 14.0  
 Nodes expanded: 4  
 Queue size: 3  
 Max queue size: 4

Con best-first, l'albero risulta:



▼ Operations

Show operations

- 1) A [f = 5.0]

- /) B [f = 1.0]
- /) G [f = 0.0]
- /) C [f = 4.0]

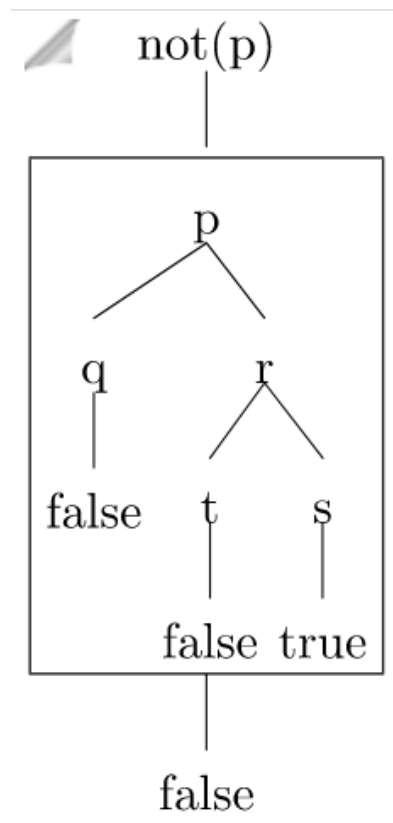
Path cost: 15.0  
 Nodes expanded: 1  
 Queue size: 2  
 Max queue size: 3

I nodi espansi sono solo A e la soluzione trovata AG non è ottimale (costo 15).

### Esercizio 6

Per la spiegazione dell'albero SLDNF per un goal negativo, si veda il materiale del corso.

L'albero SLDNF per il goal  $\text{?-not } p$  è:



E quindi  $\text{not}(p)$  fallisce perché c'è (almeno) un ramo di successo per  $p$ .