**COMPITO DI FONDAMENTI DI INTELLIGENZA ARTIFICIALE**
**INTELLIGENZA ARTIFICIALE (v.o.) – PARTE I**

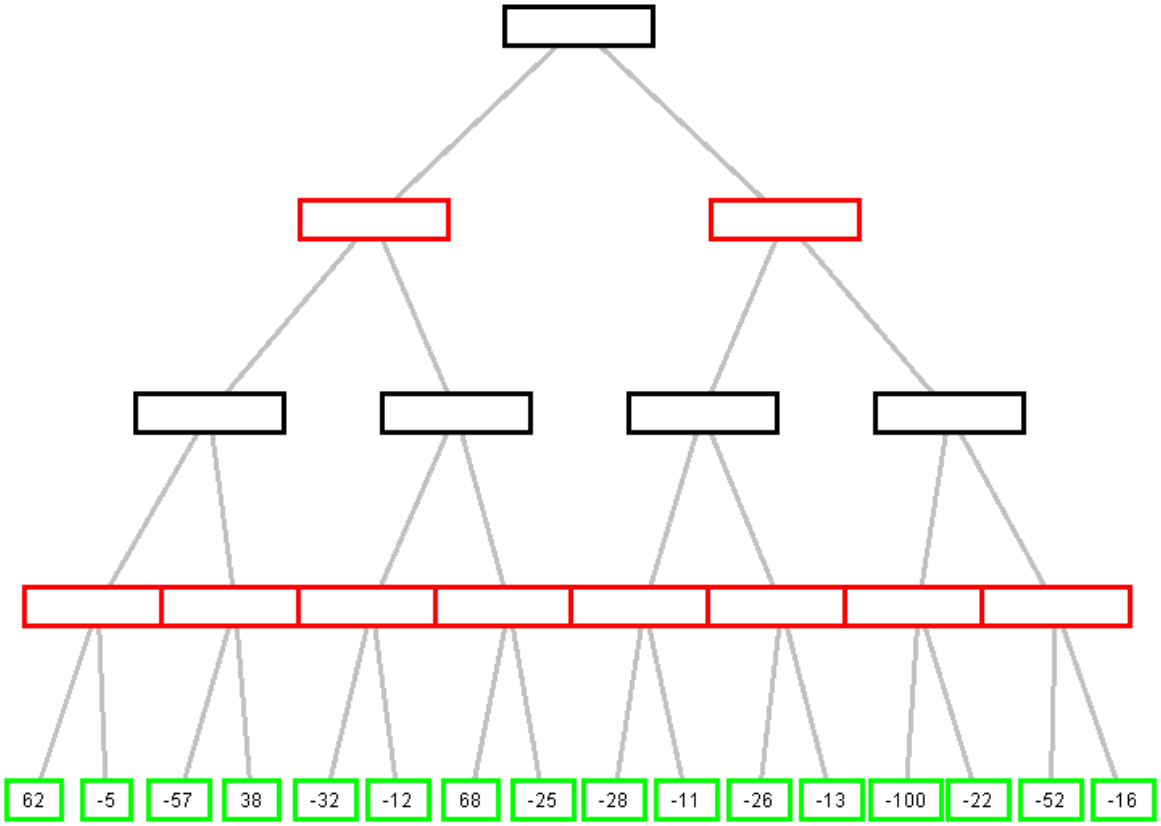
**2 Dicembre 2008 (Tempo a disposizione 2h ; su 32 punti)**

**Esercizio 1 (punti 5)**
Rappresentare le seguenti frasi:

- Due gemelli non possono appartenere allo stesso club calcistico.
- Un calciatore non può scendere in campo contro un suo gemello.
- Una persona non può appartenere al club calcistico presieduto da un suo gemello.
- Giorgio e Pasquale sono gemelli.
- Marco e Luca sono gemelli.
- Giorgio gioca nel Milan.
- Luca presiede il Bari.

in logica dei predicati del I ordine.
Dimostrare poi tramite l'applicazione del principio di risoluzione che se Pasquale appartiene al Bari, allora il Bari non può giocare contro il Milan.
N.B. Considerate "appartenere a un club" e "scendere in campo con un club" come sinonimi e utilizzate i seguenti predicati binari: gemelli/2, appartiene/2, giocacontro/2, presiede/2.

**Esercizio 2 (punti 6)**
Si consideri il seguente albero di gioco, dove i punteggi sono dal punto di vista del primo giocatore (Max):




Si mostri come l'algoritmo min-max risolve il problema. Si mostrino poi i tagli alfa-beta.

**Esercizio 3 (punti 5)**
Si consideri il seguente programma Prolog, che verifica se in una lista ci sono elementi duplicati:

```
nodup(L):- not(duplicati(L)).
duplicati(L):- membertail(X,L,T), membertail(X,T,_).
membertail(X,[X|T],T).
membertail(X,[_|L],T):- membertail(X,L,T).
```

Si mostri l'albero SLDNF relativo all'invocazione di `nodup([1,2]).`

**Esercizio 4 (punti 5)**
Si scriva un programma Prolog per un predicato `range(I,J,Ks)` che è vero se `Ks` è la lista di interi compresi tra `I` e `J` (inclusi). Si supponga che `I` e `J` siano sempre ground.
Esempi:

```
?-range(2,8,[2,3,4,5,6,7,8]).
yes
?-range(2,8,[3,2,4,5,6,7,8]).
no
?-range(2,2,L).
yes, L=[2]
```

**Esercizio 5 (punti 8)**
La figura seguente mostra un mondo artificiale in cui un agente A si trova nel quadrato (1, 1), l'obiettivo G è in posizione (1, 3) e nei quadrati (1, 2) e (2, 2) ci sono delle buche B. Se l'agente entra in un quadrato in cui c'è una buca muore. L'agente si può spostare di un quadrato alla volta in orizzontale o in verticale (in una delle quattro direzioni), ma non in diagonale e mai verso un quadrato in cui c'è una buca.

| | 1 | 2 | 3 |
|---|---|---|---|
| 3 | G | | |
| 2 | B | B | |
| 1 | A | | |

Formalizzare il problema di raggiungere l'obiettivo G come un problema di ricerca.
(a) Descrivere lo spazio degli stati.
(b) Descrivere lo stato iniziale e lo stato obiettivo.
(c) Descrivere gli operatori.
(d) Assumendo che gli operatori abbiano costo unitario, disegnare l'albero di ricerca generato dall'algoritmo A* per il problema, usando come funzione euristica la distanza di Manhattan, e determinando i valori assegnati a ciascun nodo dalla funzione di valutazione. Quando la funzione di valutazione assume lo stesso valore su più nodi, indicare un possibile ordine di espansione dei nodi stessi. Evitare di espandere gli stati già visitati.
(e) Indicare se la funzione di valutazione euristica precedente è ammissibile e perché.
Nota: questo esercizio e la sua soluzione sono simmetrici rispetto a quello dato, che era:

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | A | | |
| 2 | B | B | |
| 3 | G | | |

**Esercizio 6 (punti 3)**
Si introduca (eventualmente descrivendolo per casi) l'algoritmo di unificazione e si discuta (anche tramite esempio) se e quale proprietà di un dimostratore automatico tramite risoluzione che lo adotti è inficiata dalla rimozione del test di occur-check in tale algoritmo.

**SOLUZIONE**

**Esercizio 1**

Costanti: giorgio, pasquale, luca, marco, milan, bari; Simboli di predicato unari: club(X); Simboli di predicato binari: gemelli(X,Y), appartiene(X,Y), giocacontro(X,Y), presiede(X,Y).

Rappresentiamo le frasi in modo opportuno come formule del primo ordine:

1] $\forall X\ \forall Y\ \forall Z\ [(gemelli(X,Y) \wedge club(Z) \wedge appartiene(X, Z)) \rightarrow \neg appartiene(Y, Z)]$

2] $\forall X\ \forall Y\ \forall Z\ \forall T\ [(gemelli(X,Y) \wedge club(Z) \wedge club(T) \wedge appartiene(X, Z) \wedge appartiene(Y, T)) \rightarrow \neg giocacontro(Z, T)]$

3] $\forall X\ \forall Y\ \forall Z\ [(gemelli(X,Y) \wedge club(Z) \wedge presiede(X, Z)) \rightarrow \neg appartiene(Y, Z)]$

4] gemelli(giorgio,pasquale)

5] gemelli(marco,luca)

6] appartiene(giorgio,milan)

7] presiede(luca,bari)

8] club(milan)

9] club(bari)

Query come formula del primo ordine: Q] $appartiene(pasquale, bari) \rightarrow \neg giocacontro(bari,milan)$

Trasformazione in clausole:

1] $\neg gemelli(X,Y) \vee \neg club(Z) \vee \neg appartiene(X, Z) \vee \neg appartiene(Y, Z)$

2] $\neg gemelli(X,Y) \vee \neg club(Z) \vee \neg club(T) \vee \neg appartiene(X, Z) \vee \neg appartiene(Y, T) \vee \neg giocacontro(Z, T)$

3] $\neg gemelli(X,Y) \vee \neg club(Z) \vee \neg presiede(X, Z) \vee \neg appartiene(Y, Z)$

4] gemelli(giorgio,pasquale)

5] gemelli(marco,luca)

4’] gemelli(pasquale,giorgio)

5’] gemelli(luca,marco)

6] appartiene(giorgio,milan)

7] presiede(luca,bari)

8] club(milan)

9] club(bari)

G] $\neg (\neg appartiene(pasquale, bari) \vee \neg giocacontro(bari,milan))$

G] $appartiene(pasquale, bari) \wedge giocacontro(bari,milan)$

G1] appartiene(pasquale, bari)

G2] giocacontro(bari,milan)

Applicazione della risoluzione: da G2 e 2] si deriva:

10] $\neg gemelli(X,Y) \vee \neg club(bari) \vee \neg club(milan) \vee \neg appartiene(X, bari) \vee \neg appartiene(Y, milan)$

Da 10], 8] e 9], in più passi si deriva:

11] $\neg gemelli(X,Y) \vee \neg appartiene(X, bari) \vee \neg appartiene(Y, milan)$

Da 11], 4’] si deriva:

12] $\neg appartiene(pasquale, bari) \vee \neg appartiene(giorgio, milan)$

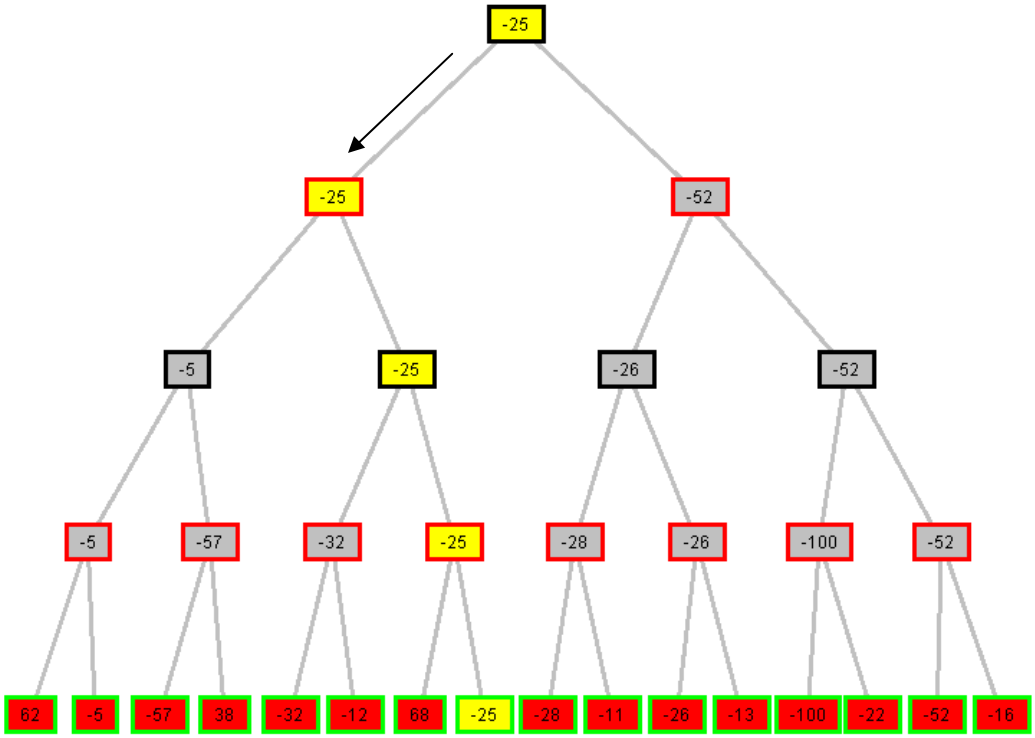
Da 12] , G1] e 6] in due passi si deriva la clausola vuota.

**Esercizio 2**

Min-Max

Theoretical minimum / actual / maximum:

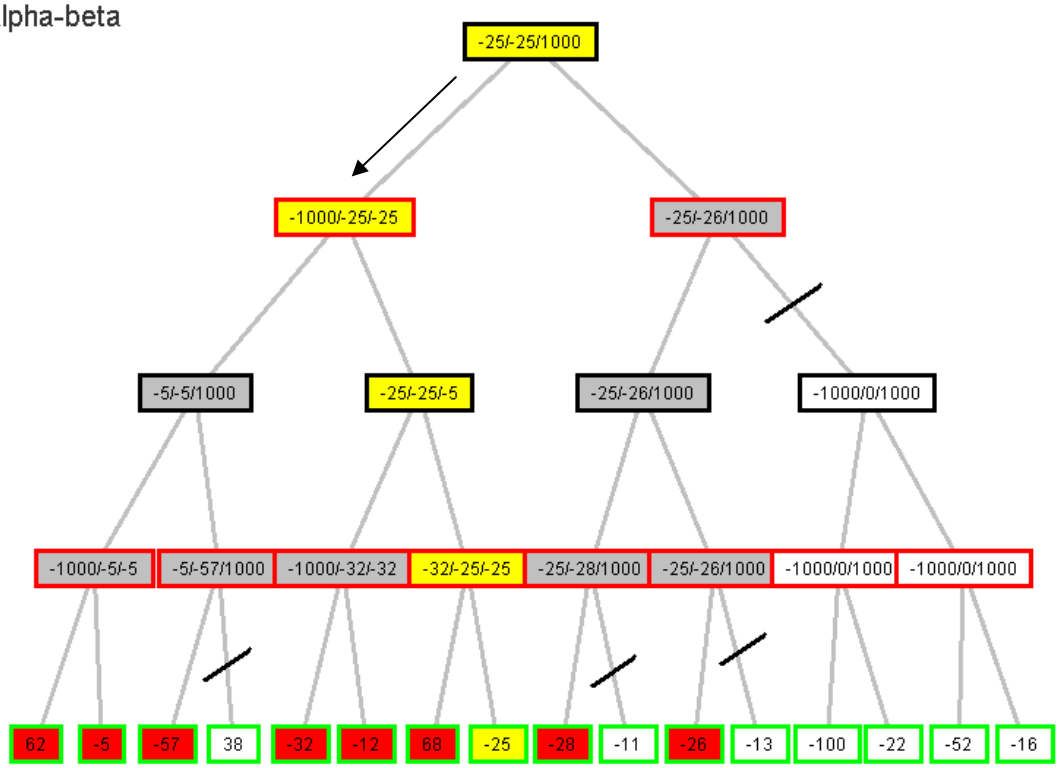
Minimax only



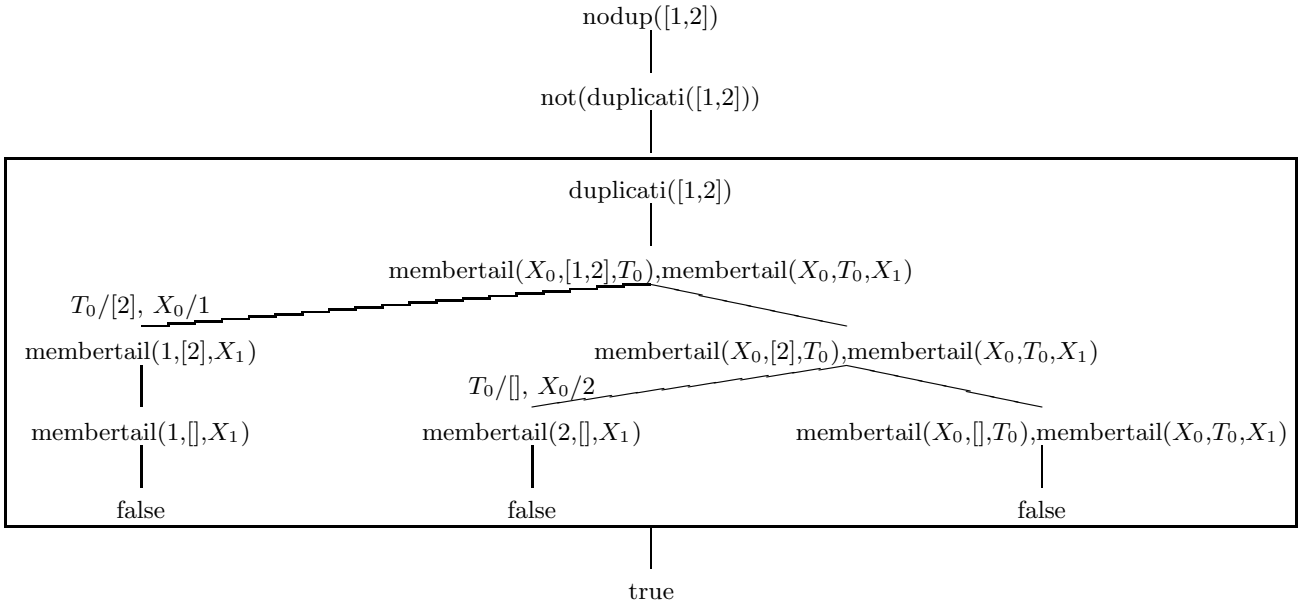

Alfa-beta

Theoretical minimum / actual / maximum:

Minimax with alpha-beta



**Esercizio 3**




**Esercizio 4**

```
/* range(I,J,Ks) e' vero se Ks e' la lista di interi compresi tra I e J     */
/*  compresi  */

range(I,J,Ks):-range(I,J,[],Ks).

range(I,I,Ks,[I|Ks]):-!.
range(I,J,As,Ks):-I < J, J1 is J - 1, range(I,J1,[J|As],Ks).
```

**Esercizio 5**

Sia N = [1,2,3]
Spazio degli stati: S = N × N
Un generico stato s sarà rappresentato da un coppia di coordinate <xs,ys> che indicano la posizione dell'agente nel mondo.
Stato Iniziale: <1,1>
Stato Finale: <1,3>
Operatori:
L'agente può muoversi di un quadrato alla volta in orizzontale o in verticale. Se vuole rimanere vivo non deve passare nei quadrati in cui c'è una buca. Sia Buca(x,y) = true se in posizione <x,y> c'è una buca.

Nome: Sopra
- Significato: A partire dalla sua posizione attuale l'agente si muove di una posizione in alto.
- Precondizioni per l'applicazione nello stato s = <xs,ys>: il passaggio non deve essere bloccato dalla presenza di una buca - Buca(xs,ys +1) = false
- Nuovo stato: s' = <xs,ys +1>

Nome: Sotto
- Significato: A partire dalla sua posizione attuale l'agente si muove di una posizione in basso.
- Precondizioni per l'applicazione nello stato s = <xs,ys>: il passaggio non deve essere bloccato dalla presenza di una buca - Buca(xs,ys -1) = false
- Nuovo stato: s' = <xs,ys +1>

Nome: Destra
- Significato: A partire dalla sua posizione attuale l'agente si muove di una posizione a destra.
- Precondizioni per l'applicazione nello stato s = <xs,ys>: il passaggio non deve essere bloccato dalla presenza di una buca - Buca(xs+1,ys) = false
- Nuovo stato: s' = <xs+1,ys>

Nome: Sinistra
- Significato: A partire dalla sua posizione attuale l'agente si muove di una posizione a sinistra.

Precondizioni per l'applicazione nello stato s = <xs,ys>: il passaggio non deve essere bloccato dalla presenza di una buca - Buca(xs-1,ys) = false

Nuovo stato: s' = <xs-1,ys>

Euristica ammissibile:

Scegliamo come euristica ammissibile h(n) la distanza di Manhattan. La distanza di Manhattan tra due punti <x1,y1> e <x2,y2> è data dalla formula M = |y2-y1| + |x2-x1|.

Algoritmo A*:

La funzione di valutazione è f(n) = g(n) + h(n), dove la h(n) è la funzione euristica precedente e la g(n) è la funzione di costo del cammino, scelta pari al numero delle caselle della griglia visitate dall'agente. L'algoritmo A* espande tra tutti i nodi aperti quello che ha il valore della funzione euristica più basso.

L'albero di ricerca è mostrato in figura. Accanto ad ogni nodo è riportato il valore della funzione di valutazione. I nodi già visitati non sono stati espansi.