FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 1° parte (6 CFU)
13 Luglio 2011 – Tempo a disposizione: 2 h – Risultato: 32/32 punti

Esercizio 1 (7 punti)
Scrivere le seguenti frasi in logica dei predicati del primo ordine:

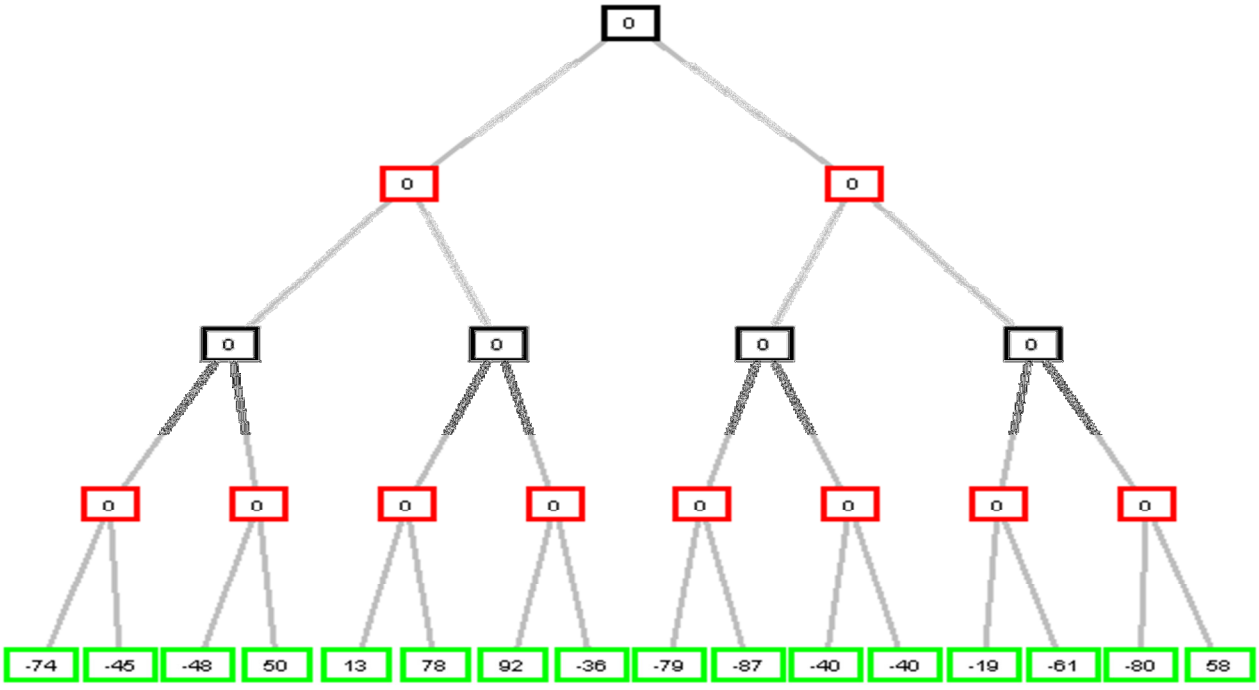
1. *Alcuni funzionari di dogana hanno perquisito tutti coloro che sono entrati nel paese, ad eccezione dei VIP.*
2. *Alcuni spacciatori di droga sono entrati nel paese e sono stati perquisiti solo da spacciatori di droga.*
3. *Nessuno spacciatore è un VIP.*
4. Quindi *alcuni funzionari sono spacciatori di droga.*

utilizzando i seguenti predicati per la loro rappresentazione:

- `e(X)` `X` è entrato nel paese,
- `v(X)` `X` è un VIP,
- `p(X,Y)` `Y` ha perquisito `X`,
- `f(X)` `X` è un funzionario di dogana,
- `s(X)` `X` è uno spacciatore di droga.

Trasformarle poi in clausole e dimostrare tramite risoluzione che dall'insieme di clausole si deriva la contraddizione.

Esercizio 2 (4 punti)
Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore (che è *Max*). Si mostri come gli algoritmi *min-max* e *alfa-beta* risolvono il problema.



Esercizio 3 (6 punti)
Disegnare gli alberi SLD che si originano dal seguente programma P con il goal

```
/* G0 */  ?- add(0, W, 0), add(0, W, K).
```

con regole di calcolo *left-most* (che seleziona l'atomo del goal più a sinistra) e *right-most* (che invece seleziona quello più a destra). Non sviluppare gli alberi oltre il livello di profondità 4, lasciando indicati eventuali rami infiniti.

```
/* P1: */ add(X, 0, X).
/* P2: */ add(X, s(Y), s(Z)) :- add(X, Y, Z).
```

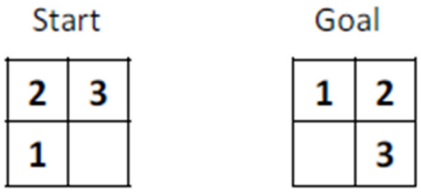
Esercizio 4 (4 punti)

Si scriva un programma Prolog che verifichi che gli elementi di una lista siano ordinati in ordine crescente.

Es.:
```
?- ordered([1, 3, 4]).
yes
?- ordered([1, 2, 3, 2, 5, 4, 1]).
no
```

Esercizio 5 (7 punti)

Si consideri il gioco del filetto con una scacchiera 2x2, nella quale si devono collocare tre tessere numerate da 1 a 3 e uno spazio vuoto. Ci sono quattro operatori che spostano lo *spazio vuoto* in alto, in basso, a sinistra, a destra di una casella. Lo *stato iniziale* ed il *goal* sono mostrati in figura:



Si mostri come raggiungere il Goal dallo stato iniziale Start, usando le seguenti strategie:

a) *Breath-first* search,
b) *Depth-first* search,
c) *A** search con *euristica* data dalla somma delle mosse fatte e delle caselle fuori posto.

Si assuma di *non* poter memorizzare gli stati già visitati. Gli operatori siano provati *nell'ordine dato*, sempre che la strategia non stabilisca un ordine diverso. Etichettare ogni nodo nell'albero espanso con un *numero progressivo* che rappresenti l'ordine di visita.
Se un metodo di ricerca non trova il risultato, motivare il perché nello svolgimento della soluzione.

Esercizio 6 (4 punti)

Si introduca la strategia di ricerca *iterative deepening*, descrivendone sinteticamente l'*algoritmo* e le *proprietà* (*complessità spaziale*, *complessità temporale*, *completezza* ed *ottimalità*).

VOTO:

- *Esame da 6 CFU, il voto è determinato da questa I parte*
- *Esame da 9 CFU, è la media pesata della I parte (che vale 2/3) e della II (che vale 1/3) ovvero il voto finale è dato da:* $\frac{2\times voto\ I\ parte + voto\ II\ parte}{3}$ *e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 2° parte (3 CFU)
13 Luglio 2011 – Tempo a disposizione: 45 min – Risultato: 32/32 punti

Esercizio 7 (8 punti)
Elencare, e descrivere brevemente i *modelli concettuali* visti a lezione per la *rappresentazione della conoscenza*, evidenziando le *differenze fondamentali* ed eventualmente fornendo qualche *esempio di base di conoscenza* che possa essere riferita a tali modelli.

Esercizio 8 (16 punti)
Una funzione importante nei *'gusci'* (*'shell'*) per sistemi esperti è quella di interfaccia con l'utente, che riguarda la funzionalità di acquisizione di dati dall'utente (*'query the user'*). Si presupponga che alcuni predicati possano essere oggetto di domande (quelli che compaiono come argomento del predicato `askable`), che le risposte ottenute siano asserite, evitando così di porre più volte le stesse domande.
Si realizzi un *meta-interprete Prolog* che implementi l'usuale risoluzione Prolog e, in più, tale funzionalità. Si abbia ad esempio il programma:

```
p :- q, r, w.
q :- s, t.
r :- u, v.
w :- s, t, u, v.
```

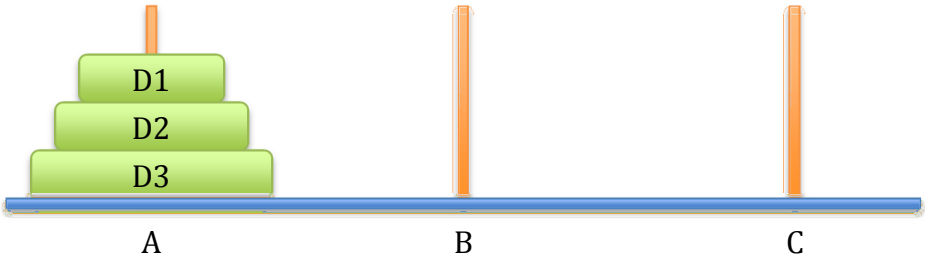
e la seguente conoscenza che definisce i predicati `s`, `t`, `u`, `v` come *'richiedibili'* all'utente:

```
askable(s).
askable(t).
askable(u).
askable(v).
```

Se invocato come segue, il meta-interprete dimostra `p` se l'utente risponde `yes` per ciascuno dei predicati `s`, `t`, `u`, `v` la prima volta che questi vengono invocati:

```
?- dim(p).
s? yes.
t? yes.
u? yes.
v? yes.
yes
```

Esercizio 9 (8 punti)
Si vuole formalizzare il problema della torre di Hanoi con tre dischi (vedi figura) attraverso il linguaggio STRIPS.



Quello che si vuole realizzare spostare i 3 dischi dal piolo A al piolo C rispettando le seguenti regole:

- si può afferrare solo un disco alla volta tra quelli presenti nella posizione superiore di ogni piolo;
- si può appoggiare un disco solo su un piolo vuoto o su un disco più grande. Perciò è necessario usare il piolo di servizio.

Detti A, B e C i tre pioli e D1, D2, D3 i tre dischi (con D1 < D2 < D3), si definisca in modo formale lo *stato iniziale*, lo *stato finale* e gli *operatori*:

- GRAB(x,y) solleva l'ultimo disco x dal piolo y,
- DROP(x,y) deposita il disco x sul piolo vuoto y,
- PICK(x,y,z) raccoglie il disco x da sopra il disco y, con x e y entrambi nel piolo z,
- POSE(x,y,z) impila il disco x sopra il disco y, se x è minore di y, con y nel piolo z,

utilizzando i seguenti predicati per definire gli stati:

- ARM_EMPTY il braccio non sta sollevando alcun disco,
- CLEAR(_) il disco indicato è in cima ad una pila di dischi,
- DISK(_) l'oggetto indicato è un disco,
- EMPTY(_) nessun disco è alloggiato sul piolo indicato,
- HOLDING(_) il braccio sta sollevando il disco indicato,
- ON(_,_) il primo disco indicato si trova sopra al secondo,
- ON_ROD(_,_) il disco indicato è alloggiato sul piolo indicato,
- ON_TABLE(_) il disco indicato è alla base di una pila di dischi,
- ROD(_) l'oggetto indicato è un piolo,
- SMALLER(_,_) il primo disco indicato è più piccolo del secondo.

VOTO:

- *Esame da 3 CFU, il voto è determinato da questa 2° parte*
- *Esame da 9 CFU, è la media pesata della 1° parte (che vale 2/3) e della 2° (che vale 1/3) ovvero il voto finale è dato da:* $\frac{2 \times voto\ I\ parte + voto\ II\ parte}{3}$ *e varia quindi da 0 ad un massimo di 32 (equivalente alla lode).*

FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 1° parte
13 Luglio 2011 – Tempo a disposizione: 2 h

Esercizio 1

Frasi in logica dei predicati del primo ordine:

1. $\forall x(E(x) \wedge \neg V(x) \rightarrow \exists y(F(y) \wedge P(x,y)))$
2. $\exists x(S(x) \wedge E(x) \wedge \forall y(P(x,y) \rightarrow S(y)))$
3. $\forall x(S(x) \rightarrow \neg V(x))$
4. $\exists x(F(x) \wedge S(x))$

Traduzione in clausole:

$$
\begin{aligned}
&1.\ \forall x(E(x) \wedge \neg V(x) \rightarrow \exists y(F(y) \wedge P(x,y))) \\
&\Rightarrow\ \forall x \exists y(\neg(E(x) \wedge \neg V(x)) \vee (F(y) \wedge P(x,y))) \\
&\Rightarrow\ \forall x(\neg E(x) \vee V(x) \vee (F(f(x)) \wedge P(x,f(x)))) \\
&\Rightarrow\ (\neg E(x) \vee V(x) \vee F(f(x))) \wedge (\neg E(x) \vee V(x) \vee P(x,f(x))) \\
&\Rightarrow\ \{\neg E(x) \vee V(x) \vee F(f(x)), \neg E(x) \vee V(x) \vee P(x,f(x))\}
\end{aligned}
$$

$$
\begin{aligned}
&2.\ \exists x(S(x) \wedge E(x) \wedge \forall y(P(x,y) \rightarrow S(y))) \\
&\Rightarrow\ \exists x \forall y(S(x) \wedge E(x) \wedge (\neg P(x,y) \vee S(y))) \\
&\Rightarrow\ S(a) \wedge E(a) \wedge (\neg P(a,y) \vee S(y)) \\
&\Rightarrow\ \{S(a), E(a), \neg P(a,y) \vee S(y)\}
\end{aligned}
$$

$$
\begin{aligned}
&3.\ \forall x(S(x) \rightarrow \neg V(x)) \\
&\Rightarrow\ \forall x(S(x) \rightarrow \neg V(x)) \\
&\Rightarrow\ \{\neg S(x) \vee \neg V(x)\}
\end{aligned}
$$

$$
\begin{aligned}
&\neg 4.\ \neg \exists x(F(x) \wedge S(x)) \\
&\Rightarrow\ \forall x \neg(F(x) \wedge S(x)) \\
&\Rightarrow\ \{\neg F(x) \vee \neg S(x)\}
\end{aligned}
$$

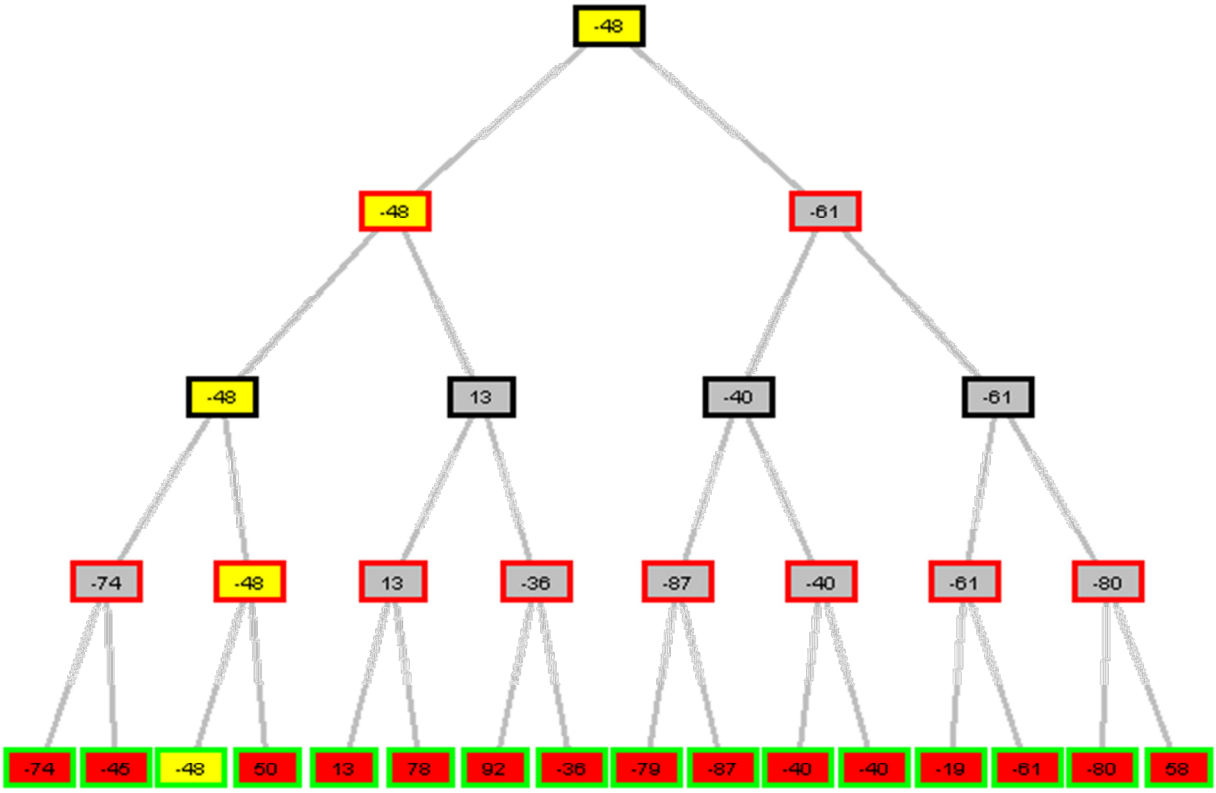
$$
\begin{aligned}
S = \{\ &\neg E(x) \vee V(x) \vee F(f(x)), \neg E(x) \vee V(x) \vee P(x,f(x)), S(a), E(a), \neg P(a,y) \vee S(y), \\
&\neg S(x) \vee \neg V(x), \neg F(x) \vee \neg S(x)\}
\end{aligned}
$$

Risoluzione (si deriva la clausola vuota dall'insieme S):

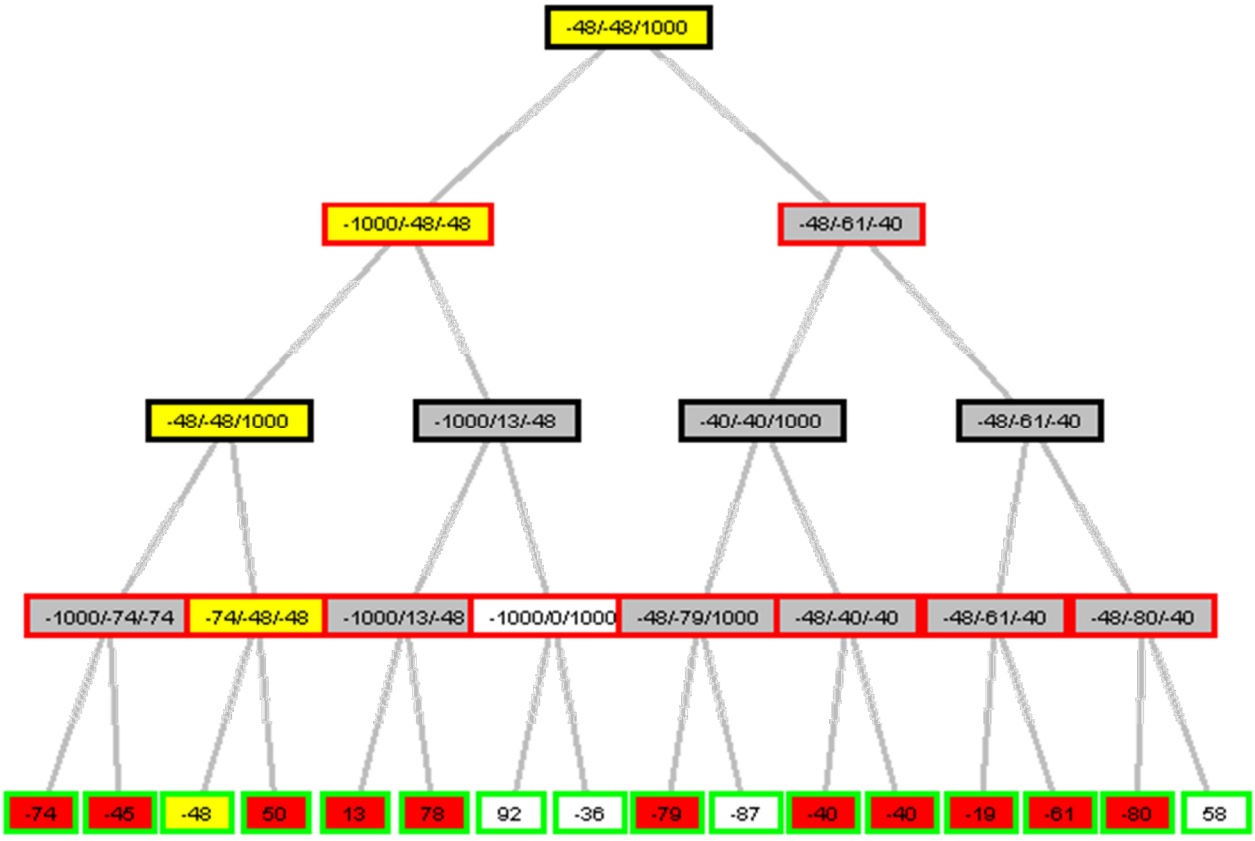
1.	$\neg E(x) \vee V(x) \vee F(f(x))$	$in\ S$
2.	$\neg E(x) \vee V(x) \vee P(x,f(x))$	$in\ S$
3.	$S(a)$	$in\ S$
4.	$E(a)$	$in\ S$
5.	$\neg P(a,y) \vee S(y)$	$in\ S$
6.	$\neg S(x) \vee \neg V(x)$	$in\ S$
7.	$\neg F(x) \vee \neg S(x)$	$in\ S$
8.	$V(a) \vee P(a,f(a))$	$Res(2,4), \{a/x\}$
9.	$\neg V(a)$	$Res(3,6), \{a/x\}$
10.	$P(a,f(a))$	$Res(8,9)$
11.	$S(f(a))$	$Res(5,10), \{f(a)/y\}$
12.	$\neg F(f(a))$	$res(7,11), \{f(a)/x\}$
13.	$V(a) \vee F((f(a))$	$Res(1,4), \{a/x\}$
14.	$F(f(a))$	$Res(9,13)$
15.	$\square$	$Res(12,14)$

Esercizio 2

min-max:

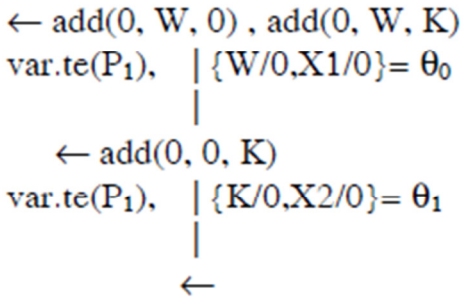


alfa-beta:



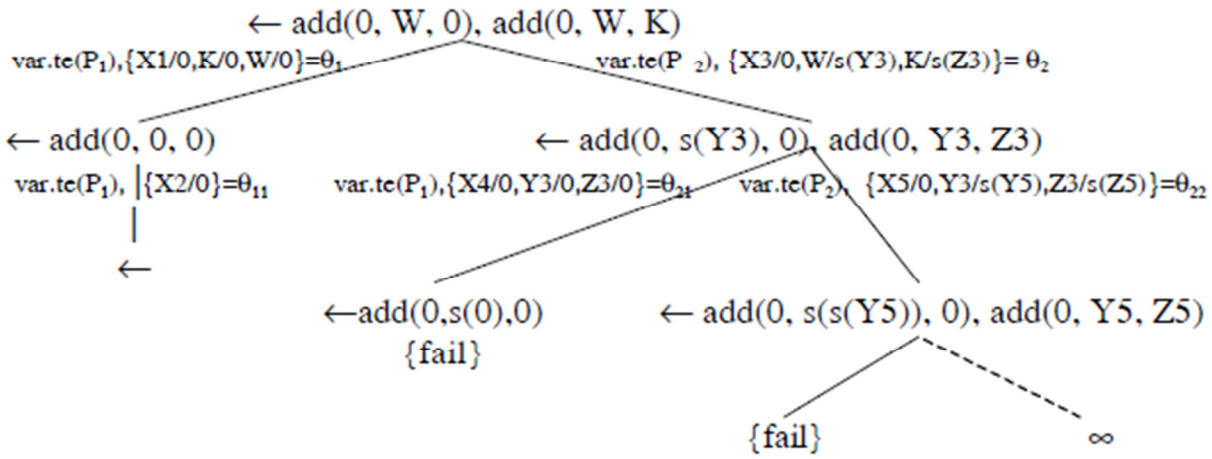
Esercizio 3

con la regola che seleziona l'atomo più a sinistra si ha il seguente albero SLD



con $\theta_0\theta_1|\ \{W,K\} = \{W/0, K/0\}$;

con la regola che seleziona l'atomo più a destra si ha il seguente albero SLD



con $\theta_1\theta_{11}|\ \{W,K\} = \{W/0, K/0\}$.

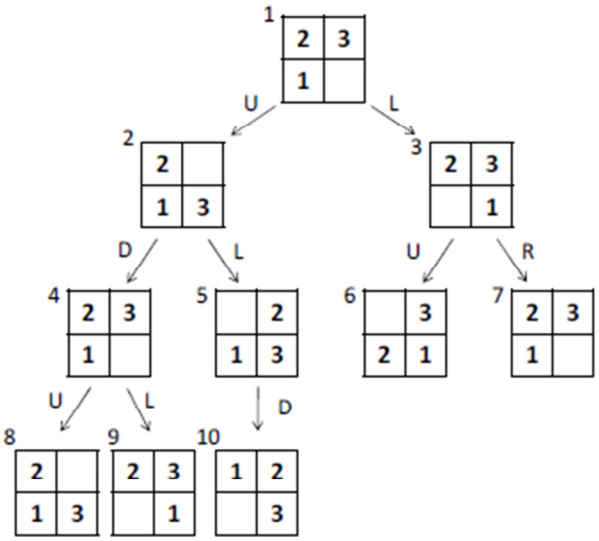
Esercizio 4

```
ordered([]).
ordered([_Single]).
ordered([Head1,Head2|Tail]) :- Head1 =< Head2, ordered([Head2|Tail]).
```

Esercizio 5 & 6

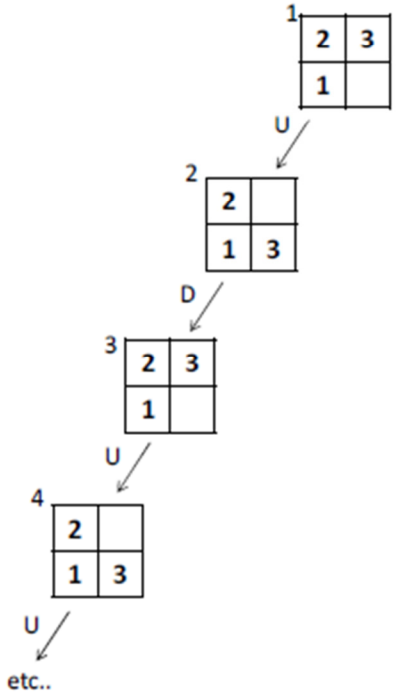
a)

Breadth first search will start from the root node, then expands all the successors of the root node, and then all their successors and so on. Breadth first search stops when first solution is found.



b)

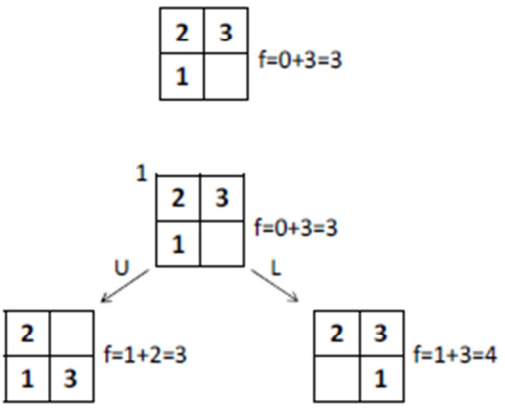
Depth-first search always expands the deepest node in the search tree. Notice that there was no mechanism to remember states that have been visited earlier. Depth first will not find a solution as it will start oscillating between movements U and D.



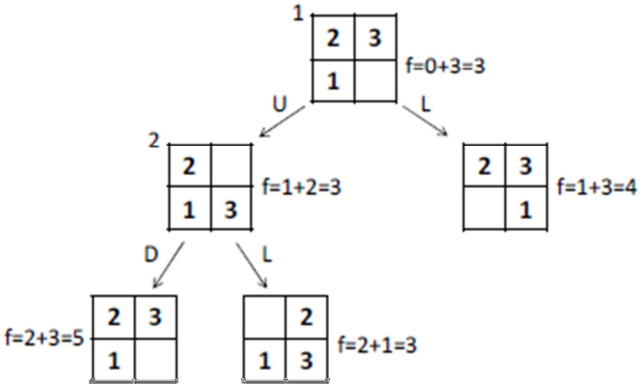
c)

A* is a search method that opens the node with smallest cost function value. The search begins from the start state.

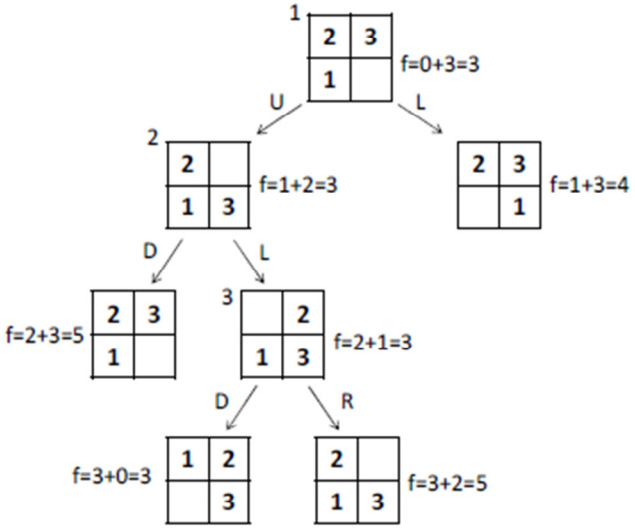
1. When the start state is opened we see two states



2. The leftmost state has a lower cost so that one is choosen and opened. We now see states with costs 5,3 and 4



3. The state with lowest cost is opened and we see two more nodes including the goal node. We notice that the goal node has the lowest cost so we choose that and can finish the search. If some other node with a lower cost function value was still visible, A* search would choose that instead of the higher cost goal node. This is because A* tries to find the path with lowest cost.



FONDAMENTI DI INTELLIGENZA ARTIFICIALE – 2° parte
13 Luglio 2011 – Tempo a disposizione: 45 min

Esercizio 7
A lezione sono stati visti le tassonomie, i tesauri, i modelli concettuali e le teorie logiche. Le tassonomie sono organizzazioni gerarchiche dove l'unica relazione tra i termini è di tipo "is-a". I tesauri, di origine linguistica, estendono il numero di relazioni possibili includendo però solo un numero ancora finito, e fissato a priori di relazioni, di origine linguistica, quali ad esempio sinonimia, iperonimia, iponimia, olonimia, meronimia, etc. I modelli concettuali estendono ulteriormente il numero di relazioni lasciando la possibilità all'utente di definire le relazioni, in numero arbitrario, ed eventualmente fornendo alcuni meccanismi di inferenza per rispondere a particolari task. Le teorie logiche infine si contraddistinguono perché supportate da una logica di riferimento sottostante (FOL) che mette a disposizione anche meccanismi d'inferenza generici.

Esercizio 8
```
dim(true) :- !.
dim((Head, Tail)) :- !, dim(Head), dim(Tail).
dim(Head) :- clause(Head, Body), dim(Body).
dim(Atom) :-
   askable(Atom), not known(Atom),
   write(Atom), write('? '),
   read(Read), nl,
   answer(Atom, Read).
known(Atom) :- istrue(Atom); isfalse(Atom).
answer(Atom, yes) :- assert(Atom), assert(istrue(Atom)).
answer(Atom, no) :- assert(isfalse(Atom)), fail.
```

Esercizio 9
Stato iniziale:

{ROD(A), ROD(B), ROD(C), DISK(D1), DISK(D2), DISK(D3), SMALLER(D1,D2), SMALLER(D2,D3), SMALLER(D1,D3), ARM_EMPTY, ON_ROD(D1,A), ON_ROD(D2,A), ON_ROD(D3,A), ON_TABLE(D3), ON(D2,D3), ON(D1,D2), CLEAR(D1), EMPTY(B), EMPTY(C)}.

Stato finale:

{ROD(A), ROD(B), ROD(C), DISK(D1), DISK(D2), DISK(D3), SMALLER(D1,D2), SMALLER(D2,D3), SMALLER(D1,D3), ARM_EMPTY, ON_ROD(D1,C), ON_ROD(D2,C), ON_ROD(D3,C), ON_TABLE(D3), ON(D2,D3), ON(D1,D2), CLEAR(D1), EMPTY(A), EMPTY(B)}.

Operatori:

- **GRAB(x,y)**
  PLIST: {ARM_EMPTY, CLEAR(x), DISK(x), ON_ROD(x,y), ON_TABLE(x), ROD(y)}.
  DLIST: {ARM_EMPTY, CLEAR(x), ON_TABLE(x), ON_ROD(x,y)}.
  ALIST: {EMPTY(y), HOLDING(x)}.

- **DROP(x,y)**
  PLIST: {DISK(x), EMPTY(y), ROD(y), HOLDING(x)}.
  DLIST: {EMPTY(y), HOLDING(x)}.
  ALIST: {ARM_EMPTY, CLEAR(x), ON_TABLE(x), ON_ROD(x,y)}.

- **PICK(x,y,z)**
  PLIST: {ARM_EMPTY, CLEAR(x), DISK(x), DISK(y), ON(x,y), ON_ROD(x,z), ON_ROD(y,z), ROD(z) , SMALLER(x,y)}.
  DLIST: {ARM_EMPTY, CLEAR(x), ON(x,y), ON_ROD(x,z)}.
  ALIST: {CLEAR(y), HOLDING(x)}.

- **POSE(x,y,z)**
  PLIST: {HOLDING(x), CLEAR(y), DISK(x), DISK(y), ON_ROD(y,z), ROD(z), SMALLER(x,y)}.
  DLIST: {CLEAR(y), HOLDING(x)}.
  ALIST: {ARM_EMPTY, CLEAR(x), ON(x,y), ON_ROD(x,z)}.