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

13 Dicembre 2006 (Tempo a disposizione 2h; su 32 punti)

Esercizio 1: (punti 6)
Sia data una formulazione CSP del problema delle 6 regine in cui ogni variabile V_i (i =1, ... 6) può assumere come valori la posizione della regine all'interno della i-esima colonna.
Si parte avendo già collocato la prima regina come in figura (V_1= 4).

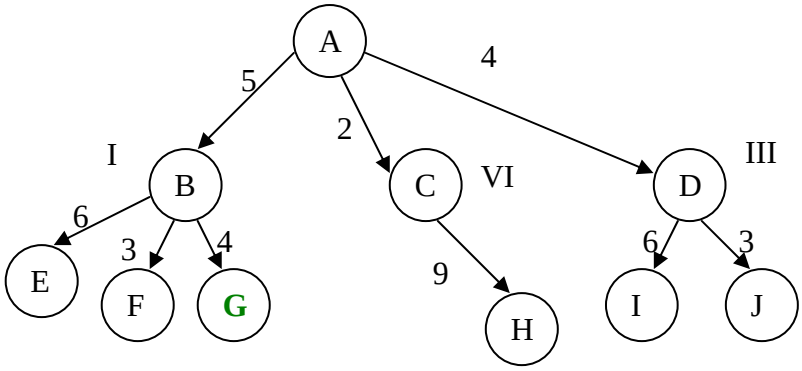
	1	2	3	4	5	6
1	⊗			⊗		
2	⊗		⊗			
3	⊗	⊗				
4	♛	⊗	⊗	⊗	⊗	⊗
5	⊗	⊗				
6	⊗		⊗			

Si mostri l'evoluzione dell'algoritmo CSP con le euristiche definite sotto, fino al momento in cui si trova la soluzione o si è costretti a fare backtracking. A parità di euristica si scelga la variabile di indice inferiore e il valore più alto.

a. FFP (First Fail Principle) in combinazione con Forward checking (FC)
b. FFP + FC + scelta del valore meno vincolante

Si precisa che per "scelta del valore meno vincolante" si intende il valore per la variabile che esclude meno valori per le altre variabili direttamente collegate con vincoli.

Esercizio 2 (punti 7)
Si consideri il seguente albero di ricerca in cui i numeri (arabi) sugli archi sono i costi e quelli (romani) vicino agli stati le stime euristiche *h*. Ove *h* non è specificato la si assuma uguale a 0.



Si assuma che i nodi siano espansi in ordine alfabetico, nel caso in cui la strategia di ricerca applicata li consideri equivalenti. Il goal sia lo stato G e la ricerca termini al suo raggiungimento.
Si indichi la lista di stati espansi da ognuna delle seguenti strategie di ricerca:

- Breadth First
- Depth First
- Iterative Deepening Search *(facoltativo)*
- Best-First search
- A* search

Si indichi inoltre se la funzione *h* è ammissibile in questo esempio. E' consistente? Si motivi la risposta data.

Esercizio 3 (punti 4)
Si scriva un programma Prolog `listediff(L1,L2,Lout)` che date in ingresso due liste `L1`, `L2` produca in uscita una lista `Lout` differenza delle due. La lista `Lout` deve essere la sequenza `L1` privata della sequenza (finale) `L2`.

Esempi:

```
?-listediff([a,b,c,d], [c,d], X)
restituisce
yes   X=[a,b]

?- listediff([a,b,c,d], [d,c], X)
fallisce

?-listediff([a,b], [a,b], X)
restituisce
yes   X=[]

?-listediff([[2,3],[1,2],a], [[1,2],a], X)
restituisce
yes   X=[[2,3]]
```

Esercizio 4 (punti 7)

Si traducano le seguenti frasi nella logica dei predicati del primo ordine, poi in forma a clausole:

- Le persone si dividono in uomini e donne
- Gli uomini e le donne sono persone
- Un genitore è una persona che è padre o madre di qualcuno; viceversa, ogni persona che è padre o madre di qualcuno ne è genitore
- Un nonno è un uomo che è padre di qualcuno che è a sua volta genitore
- Anna è una donna
- Luca, Giuseppe e Piero sono uomini
- Anna è madre di Luca
- Piero è padre di Anna
- Tweety non è una persona

Si usi poi il principio di risoluzione per dimostrare che esiste almeno un nonno di Luca.

Esercizio 5 (punti 6)

Considerando il seguente programma Prolog che ricerca elementi in alberi binari di ricerca rappresentati da strutture `t/3` a tre argomenti: `t(sottoalbero_sinistro, radice, sottoalbero_destro)`, dove il primo e il terzo argomento possono a loro volta essere alberi binari (la costante `empty/0` rappresenti un albero vuoto):

```
bsearch (El, t(Left, El, Right)):-!.
bsearch (El, t(Left, Root, Right)):-
                          El<Root, bsearch(El, Left).
bsearch (El, t(Left, Root, Right)):- El>Root, bsearch(El, Right).
```

Si rappresenti l'albero di derivazione SLDNF relativo al goal

```
?-bsearch(5, t(empty, 3, t( t(empty,5,empty), 7, t(empty, 9, empty))) ).
```

e si dica qual è la risposta calcolata.

Si discuta inoltre come potrebbe essere ottimizzata la computazione, limitandone il non-determinismo attraverso l'utilizzo del predicato cut (indicare sul testo le modifiche).

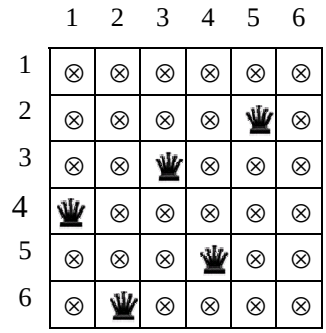
Esercizio 6 (punti 2)

Si precisi cosa si intende con Skolemizzazione, quando e come si applica, anche aiutandosi con esempi.

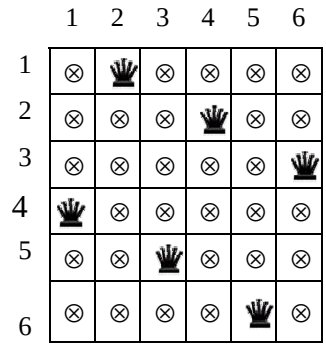
SOLUZIONE

Esercizio 1:

a.

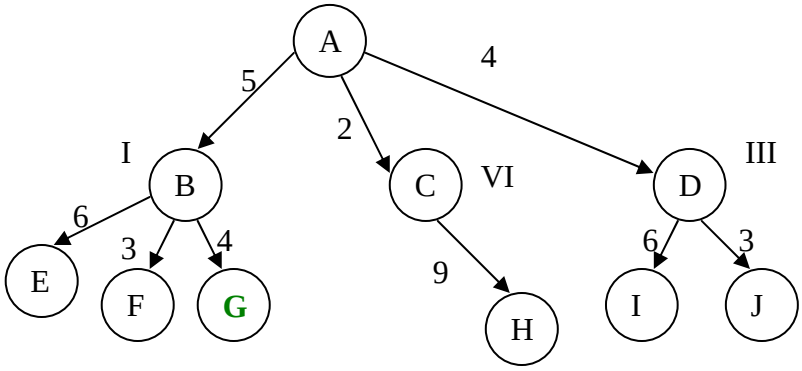


V1=4 V2={1, 2, 6} V3={1, 3, 5} V4={2, 3, 5, 6} V5={1, 2, 3, 5, 6} V6={1, 2, 3, 5, 6}	V1=4 V2=6 V3={1, 3} V4={2, 3, 5} V5={1, 2, 5} V6={1, 3, 5}	V1=4 V2=6 V3=3 V4={5} V5={2} V6={1, 5}	V1=4 V2=6 V3=3 V4=5 V5={2} V6={1}	V1=4 V2=6 V3=3 V4=5 V5=2 V6={ }



V1=4 V2={1[5], 2[7], 6[6]} V3={1, 3, 5} V4={2, 3, 5, 6} V5={1, 2, 3, 5, 6} V6={1, 2, 3, 5, 6}	V1=4 V2=1 V3={3[5], 5[5]} V4={2, 5, 6} V5={2, 3, 5, 6} V6={2, 3, 6}	V1=4 V2=1 V3=5 V4={2} V5={2, 6} V6={3, 6}	V1=4 V2=1 V3=5 V4=2 V5={6} V6={3, 6}	V1=4 V2=1 V3=5 V4=2 V5=6 V6={3}	V1=4 V2=1 V3=5 V4=2 V5=6 V6=3

Esercizio 2:



Breadth First	A B C D E F G
Depth First	A B E F G
Iterative Deepening Search	A A B C D A B E F G
Best-First search	A B E F G
A* search	A B D J C F G

La funzione *h* è ammissibile in questo esempio perché sempre minore del valore costo effettivo. E' anche consistente inoltre perché per ogni nodo $h(n) <= c(n,a,n') + h(n')$.

Esercizio 3:

```
listediff(L1,L2,L3):- append(L3,L2,L1).

append([],L,L):-!.
append([H|T],L,[H|T1]):- append(T,L,T1).
```

Esercizio 4:

Logica:

- Le persone si dividono in uomini e donne

$$\forall X(persona(X) \Rightarrow (uomo(X) \text{ xor } donna(X))$$

- Gli uomini e le donne sono persone

$$\forall X(uomo(X) \vee donna(X) \Rightarrow persona(X))$$

- Un genitore è una persona che è padre o madre di qualcuno; viceversa, ogni persona che è padre o madre ne è genitore

$$\forall XY(genitore(X,Y) \Leftrightarrow (padre(X,Y) \vee madre(X,Y)) \wedge persona(X))$$

- Un nonno è un uomo che è padre di qualcuno che è a sua volta genitore di qualcun altro

$$\forall X \forall Y \ \forall Z \ (uomo(X), padre(X,Y), genitore(Y, Z) \Rightarrow nonno(X,Z))$$

- Anna è una donna

$$donna(anna)$$

- Luca, Giuseppe e Piero sono uomini

$$uomo(luca)$$
$$uomo(giuseppe)$$
$$uomo(piero)$$

- Anna è madre di Luca

  $madre(anna,luca)$
- Piero è padre di Anna

  $padre(piero,anna)$
- Tweety non è una persona

  $\neg persona(tweety)$

Query: $\exists X\ nonno(X,luca)$

Goal: $\forall X\ \neg nonno(X,luca)$

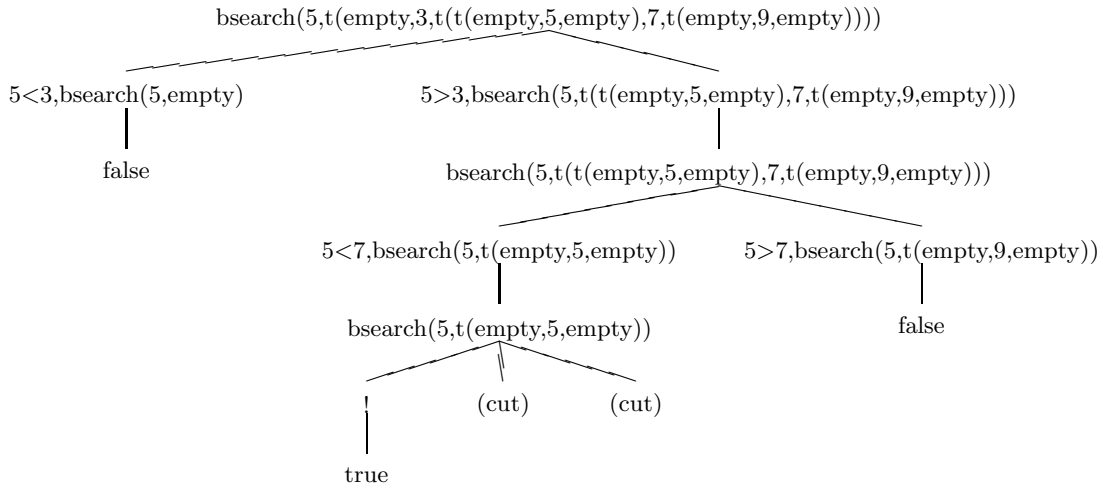
Trasformazione in clausole

- $\neg persona(X) \vee uomo(X) \vee donna(X)$
- $\neg persona(X) \vee \neg donna(X) \vee \neg uomo(X)$
- $\neg uomo(X) \vee persona(X)$
- $\neg donna(X) \vee persona(X)$
- $\neg genitore(X,Y) \vee padre(X,Y) \vee madre(X,Y)$
- $\neg genitore(X,Y) \vee persona(X)$
- $genitore(X,Y) \vee \neg padre(X,Y) \vee \neg persona(X)$
- $genitore(X,Y) \vee \neg madre(X,Y) \vee \neg persona(X)$
- $\neg uomo(X) \vee \neg padre(X,Y) \vee \neg genitore(Y,Z) \vee nonno(X,Z)$
- $donna(anna)$
- $uomo(luca)$
- $uomo(giuseppe)$
- $uomo(piero)$
- $madre(anna,luca)$
- $padre(piero,anna)$
- $\neg persona(tweety)$
- $\neg nonno(X,luca)$

Risoluzione

1. $\neg donna(A) \vee persona(A)$
2. $genitore(A,B) \vee \neg madre(A,B) \vee \neg persona(A)$
3. $\neg uomo(A) \vee \neg padre(A,B) \vee \neg genitore(B,C) \vee nonno(A,C)$
4. $\neg nonno(A,luca)$
5. $donna(anna)$
6. $uomo(piero)$
7. $madre(anna,luca)$
8. $padre(piero,anna)$
9. (da , **Errore. L'origine riferimento non è stata trovata.**) $persona(anna)$
10. (da , ,) $genitore(anna,luca)$
11. (da , , ,) $nonno(piero,luca)$
12. (da ,) []

Esercizio 5:



Si può inserire un cut dopo che è stato identificato che l'elemento si trova nel sottoalbero di sinistra (se è presente):

```
bsearch (El, t(Left, El, Right)):-!.
bsearch (El, t(Left, Root, Right)):-
                        El<Root, !, bsearch(El, Left).
bsearch (El, t(Left, Root, Right)):- El>Root, bsearch(El, Right).
```

In questo caso l'albero viene modificato come segue: