**FONDAMENTI DI INTELLIGENZA ARTIFICIALE**

**11 Febbraio 2010 – Tempo a disposizione 2h – Risultato 32/32 punti**

**Esercizio 1 (punti 6)**

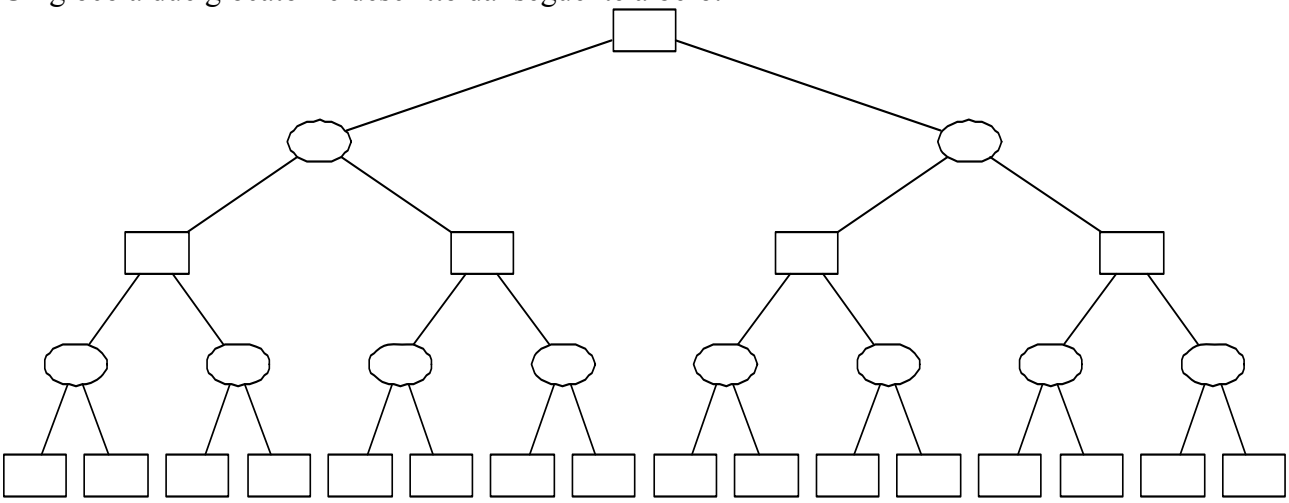
Si consideri un robot che trasporta pacchi tra le stanze di un ufficio e si supponga che la sua base di conoscenza contenga le seguenti formule, dove p(X) indica che X è un pacco, l(X,S) che il passo X è nella stanza S e s(X,Y) che il pacco X è più piccolo del pacco Y

$\forall X \forall Y$ p(X) and p(Y) and l(X,27) and l (Y,28) --> s(X,Y)
p(a)
p(b)
l(a,27) $\vee$ l(a,28)
l(b,27)
$\neg$ s(b,a)

A partire da tale base di conoscenza, la si trasformi in clausole e, applicando il principio di risoluzione, si dimostri che il pacco *a* è nella stanza 27.

**Esercizio 2 (punti 5)**

Un gioco a due giocatori è descritto dal seguente albero:




Dove i punteggi sono tutti 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 6)**

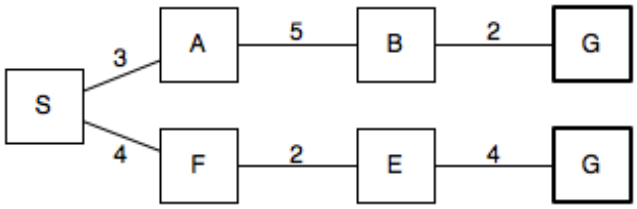
Il seguente programma Prolog risolve un problema di gioco da punto di vista del giocatore *min*.

```
minmax(min,win).
minmax(max,lose).
minmax(P,tree(L,R)):- changePlayer(P,NP),
    not((minmax(NP,L), minmax(NP,R))).
changePlayer(min,max).
changePlayer(max,min).
```

Descrive con termini *tree/2* un albero binario; le foglie possono essere *win* o *lose* (a seconda se in quella configurazione, il giocatore *min* vince o perde). Si mostri l'albero di derivazione SLDNF relativo al goal: `:- minmax(min,tree(tree(win,lose),tree(win,win))).`

**Esercizio 4 (punti 6)**

Si consideri il seguente grafo che rappresenta uno spazio di stati. S è lo stato iniziale, G lo stato goal. Gli archi sono etichettati con i costi.




A parità di euristica si utilizzi l'ordine alfabetico. Quando serve, si usi la seguente funzione di valutazione euristica $h(n)$, una stima della distanza del nodo $n$ dal goal

| $n$ | S | A | B | E | F | G |
|---|---|---|---|---|---|---|
| $h(n)$ | 8 | 7 | 2 | 4 | 5 | 0 |

Per ognuno dei seguenti algoritmi si mostri l'ordine in cui i nodi vengono espansi e i cammini soluzione trovati dai seguenti algoritmi di ricerca:

a. Ricerca di costo uniforme
b. Ricerca greedy best-first
c. Ricerca A*

**Esercizio 5 (punti 5)**

Si scriva un predicato Prolog `subset(Xs,Ys)`che ha successo se la lista `Xs` è un sottoinsieme proprio della lista `Ys` (ogni elemento di `Xs` compare in `Ys` e `Ys` contiene almeno un elemento che non compare in `Xs`). Esempi:

```
?-subset([1,2],[3,2,1]).
Yes
?-subset([1,2],[ 2,1]).
No
?-subset([3,1,2],[ 2,1]).
No
```

**Esercizio 6 (punti 4)**

Si discutano le tecniche di propagazione di vincoli. Se ne mostrino i vantaggi ottenibili sull'esempio:

$A>B$
$A>C$
$B>C$

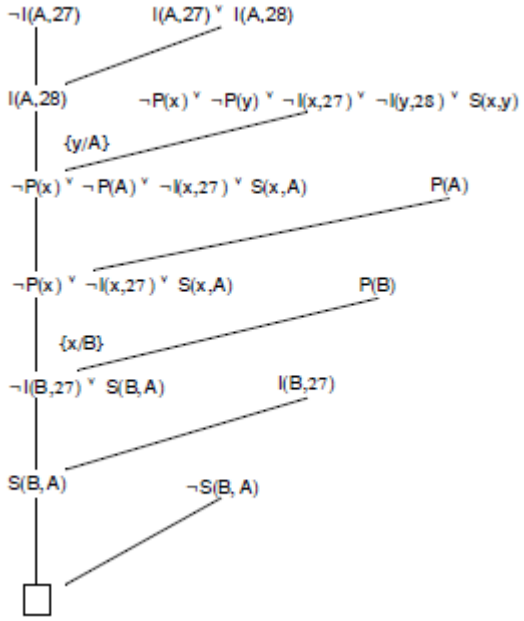
dove le variabili hanno dominio $A,B,C$::[1,2,3].

In particolare si mostri l'albero di ricerca nel caso di standard backtracking e si discuta come esso varia nel caso di applicazione del forward checking (applicato dopo almeno un *labeling*). Nello sviluppo dell'albero si considerino le variabili in ordine alfabetico e i valori del dominio in ordine crescente.

**SOLUZIONE**

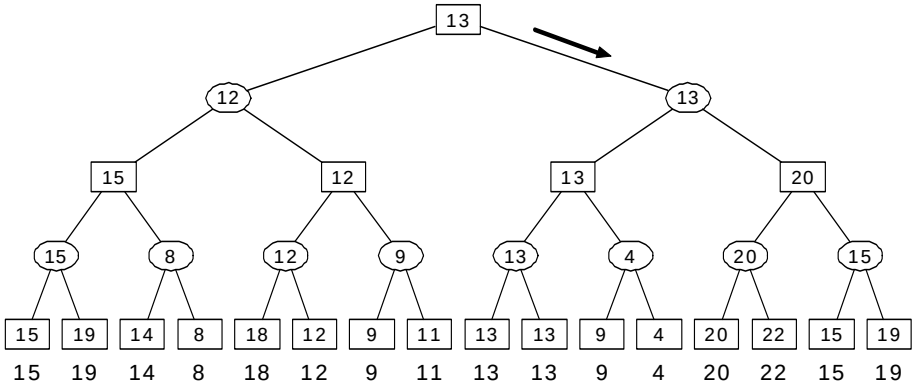
## Esercizio 1

(nella soluzione i predicati e le costanti sono con iniziale maiuscola, le variabili in minuscolo)

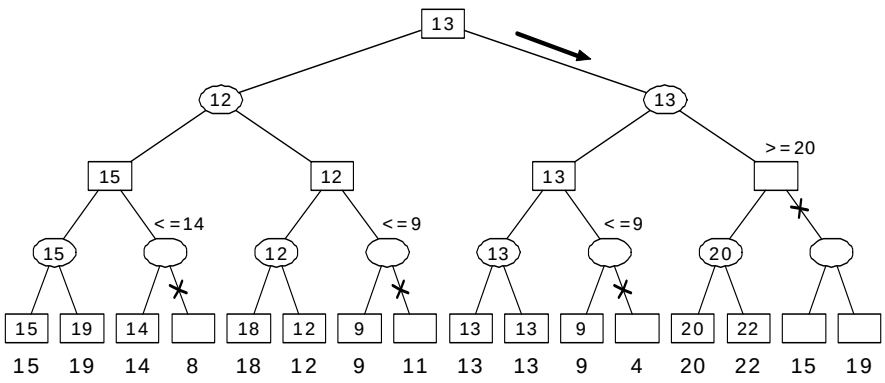



## Esercizio 2

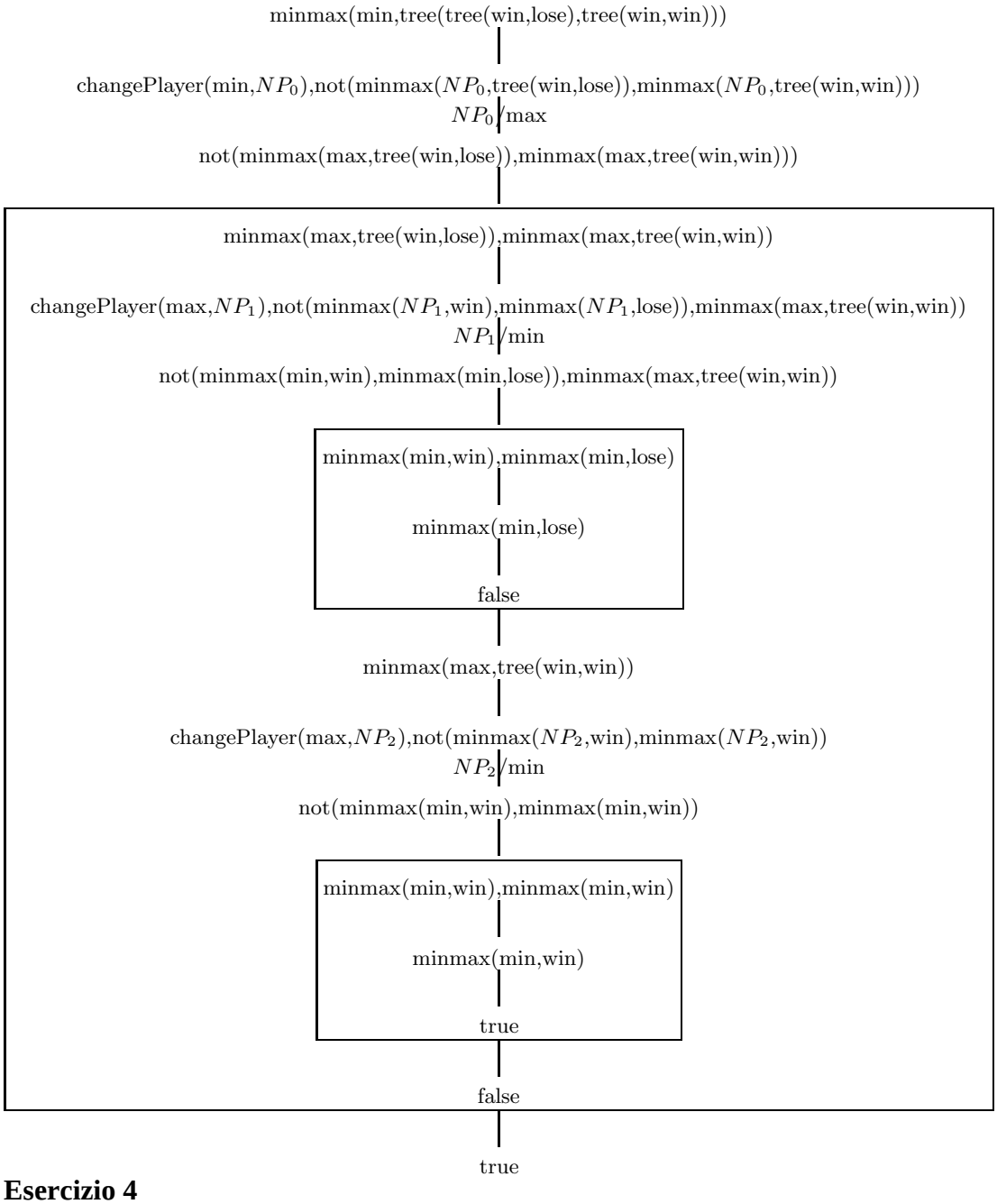
min-max:




alfa-beta:



**Esercizio 3**




**Esercizio 4**

Ricerca di costo uniforme

Ordine di espansione: S A F E B G
Cammino di soluzione: S F E G

Ricerca greedy best-first

Ordine di espansione: S, F, E, G
Cammino soluzione: S, F, E, G

Ricerca A*

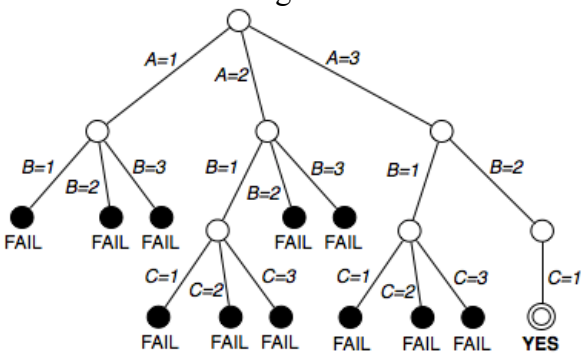
Ordine di espansione: S, F, A, B, E, G
Cammino soluzione: S, F, E, G

**Esercizio 5**

```
% subset(Xs,Ys) <- every element of list Xs occurs in list Ys
subset(Xs,Ys):-subseteq(Xs,Ys),
               member(A,Ys),
               not member(A,Xs).
subseteq([],_).
subseteq([X|Xs],Ys):-
        member(X,Ys),
        subset(Xs,Ys).
```

**Esercizio 6**

Standard backtracking:




Forward checking: