# Modalità di esame per il corso di Programmazione Avanzata a partire dall'a.a. 2009/2010

L'esame consiste nella presentazione e la discussione di un progetto in Ocaml.
Il progetto può essere lo stesso presentato per il modulo di Intelligenza Artificiale oppure, nel caso in cui questo venga sviluppato con strumenti diversi, può essere scelto nella lista sottostante.

# Modalità di esame per il corso di Programmazione Avanzata a.a. 2007/2008

L'esame consiste in una prova scritta (che si svolgerà con la modalità di una prova pratica al calcolatore e prevede esercizi su liste) e nella presentazione di un progetto.

# Progetti di esame per il corso di Programmazione Avanzata

Dovrà essere presentata una breve relazione sul progetto che illustri in particolare le strutture dati utilizzate.
Inoltre vanno preparati alcuni esempi di applicazione della funzione sviluppata (minimo due).
I progetti vanno sviluppati individualmente, tranne quelli esplicitamente indicati per coppie.
Prima di iniziare il lavoro è opportuno discutere col docente i dettagli delle specifiche del progetto scelto.
**Nota bene:** il codice del progetto dovrà essere opportunamente commentato; i nomi di funzioni e variabili e i commenti dovranno essere significativi e tassativamente in italiano.

### Elenco dei progetti

**orario ferroviario (ricerca branch and bound)**
Si consideri un grafo orientato che rappresenta una rete ferroviaria. I vertici rappresentano le stazioni. Gli archi sono etichettati con una lista di orari di partenza e di orari di arrivo.

Si scriva un programma che dati una stazione ed un orario di partenza ed una stazione di arrivo restituisca il percorso di durata minima.
Si risolva il problema utilizzando un algoritmo di ricerca branch and bound.
Come funzione di valutazione si usi la durata del cammino parziale.

**colorazione grafo (ricerca ampiezza)**
dato un grafo ed un insieme di "colori", cioè di interi da 1 ad N, assegnare un intero ad ogni vertice in modo che a vertici adiacenti siano assegnati colori diversi.
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**colorazione grafo (ricerca profondità)**
dato un grafo ed un insieme di "colori" cioè di interi da 1 ad N assegnare un intero ad ogni vertice in modo che a vertici adiacenti siano assegnati colori diversi.
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**problema commesso viaggiatore ripetente (ricerca branch and bound)**
Data una rete di città connesse tramite delle strade (rappresentata tramite un grafo con archi pesati), trovare il percorso di minore distanza che un commesso viaggiatore deve seguire per visitare tutte le città.
(Nel problema classico si richiede che ogni città si visitata una e una sola volta.)
Si risolva il problema utilizzando un algoritmo di ricerca branch and bound.
Come funzione di valutazione si usi la lunghezza del cammino parziale.

**problema salto del cavallo (ricerca ampiezza)**
Data una scacchiera NxN ed un cavallo posizionato su una casella trovare una sequenza di mosse che consenta al cavallo di occupare tutte le caselle delle scacchiera ciascuna esattamente una volta prima di ritornare alla posizione di partenza.
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**problema salto del cavallo (ricerca profondità)**
Data una scacchiera NxN ed un cavallo posizionato su una casella trovare una sequenza di mosse che consenta al cavallo di occupare tutte le caselle delle scacchiera ciascuna esattamente una volta prima di ritornare alla posizione di partenza.
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**problema dello zaino (ricerca branch and bound)**
Si consideri uno zaino che può supportare un determinato peso P e una serie di N oggetti. Ogni oggetto ha un peso e fornisce un'utilità ( ovvero un guadagno).
Si scriva un programma che, dati P, N e la descrizione degli oggetti, inserisca oggetti nello zaino in modo da massimizzare l'utilità senza eccedere il peso P.

Si risolva il problema utilizzando un algoritmo di ricerca branch and bound.
Come funzione di valutazione si usi l'utilità della soluzione parziale.

**problema della numerazione graziosa (graceful numbering) (ricerca profondità)**
Dato un grafo non orientato G=(V,E) determinare, se possibile una numerazione dei vertici tale che: $\{|Label(u)-Label(w)| : (u,w) \in E\}=\{1, 2, \dots , |E|\}$.
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**problema della numerazione graziosa (graceful numbering) (ricerca ampiezza)**
Dato un grafo non orientato G=(V,E) determinare, se possibile una numerazione dei vertici tale che: $\{|Label(u)-Label(w)| : (u,w) \in E\}=\{1, 2, \dots , |E|\}$.
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**problema della cricca (ricerca profondità)**
Una cricca in un grafo non orientato G è un insieme V di vertici tale che, per ogni coppia di vertici in V, esiste un arco che li collega. Equivalentemente, si potrebbe dire che il sottografo indotto da V è un grafo completo. La dimensione di una cricca è il numero di vertici che contiene.
Si scriva un programma che, dato un grafo ed un intero N , restituisca, se esiste, una cricca di dimensione N.
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**problema della cricca (ricerca ampiezza)**
Una cricca in un grafo non orientato G è un insieme V di vertici tale che, per ogni coppia di vertici in V, esiste un arco che li collega. Equivalentemente, si potrebbe dire che il sottografo indotto da V è un grafo completo. La dimensione di una cricca è il numero di vertici che contiene.
Si scriva un programma che, dato un grafo ed un intero N , restituisca, se esiste, una cricca di dimensione N.
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**problema della soddisfacibilità (ricerca ampiezza)**
Data una formula espressa in forma congiuntiva normale, determinare, se esiste un assegnamento delle variabili che la rende vera.
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**problema della soddisfacibilità (ricerca profondità)**
Data una formula espressa in forma congiuntiva normale, determinare, se esiste un assegnamento delle variabili che la rende vera.
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**problema della soddisfacibilità (best-first)**

Data una formula espressa in forma congiuntiva normale, determinare, se esiste un assegnamento delle variabili che la rende vera.
Si risolva il problema utilizzando un algoritmo di ricerca best-first.
Come funzione di valutazione si può il numero di clausole rese vere dalla soluzione parziale.

**problema della soddisfacibilità (hill climbing)**
Data una formula espressa in forma congiuntiva normale, determinare, se esiste un assegnamento delle variabili che la rende vera.
Si risolva il problema utilizzando un algoritmo di ricerca hill climbing.
Come funzione di valutazione si può il numero di clausole rese vere dalla soluzione parziale.

**Soluzione gioco del 15 (ricerca profondità)**
Si scriva un programma che data una configurazione del gioco del 15 determini, se esiste, una sequenza di mosse che porti alla soluzione
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**Soluzione gioco del 15 (ricerca ampiezza)**
Si scriva un programma che data una configurazione del gioco del 15 determini, se esiste, una sequenza di mosse che porti alla soluzione
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**Soluzione gioco dei cubi (ricerca profondità)**
Sia dato un insieme C di N colori ed un insieme Q di N cubi tali che ogni faccia di ciascun cubo è colorata con un colore di C.
Trovare, se esiste, una disposizione degli N cubi tale che ognuno degli N colori appare in ciascuno dei 4 lati della colonna
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**Soluzione gioco dei cubi (ricerca ampiezza)**
Sia dato un insieme C di N colori ed un insieme Q di N cubi tali che ogni faccia di ciascun cubo è colorata con un colore di C.
Trovare, se esiste, una disposizione degli N cubi tale che ognuno degli N colori appare in ciascuno dei 4 lati della colonna
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**Problema di inframezzamento (ricerca profondità)**
Dato un insieme A ed una collezione C di terne ordinate di elementi distinti di A, trovase se esiste una funzione biunivoca tale F: $A \rightarrow \{1,2,\ldots,|A|\}$ tale che per ogni terna $(a,b,c) \in C$ $f(a)<f(b)<f(c)$ oppure $f(a)>f(b)>f(c)$?
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**Problema di inframezzamento (i reggitori di moccoli) (ricerca ampiezza)**

Dato un insieme A ed una collezione C di terne ordinate di elementi distinti di A, trovase se esiste una funzione biunivoca tale F: A $\rightarrow$ {1,2,…,|A|} tale che per ogni terna (a,b,c) $\in$ C f(a)<f(b)<f(c) oppure f(a)>f(b)>f(c)?
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**Scheduling in ambiente multiprocessore (ricerca profondità)**
Siano T un insieme di task ciascuno di durata l(t), m il numero di processori P, D un tempo massimo.
Si determini, se esiste, un assegnamento dei task ai processori che rispetti il tempo massimo, cioè una funzione A : T $\rightarrow$ IN (dai task al momento di inizio) tale che per ogni u >= 0 il numero dei task t per i quali A(t) <= u < A(t) + l(t) è minore o uguale a m e per ogni t, A(t) + l(t) <= D.
Si risolva il problema utilizzando un algoritmo di ricerca in profondità.

**Scheduling in ambiente multiprocessore (ricerca ampiezza)**
Siano T un insieme di task ciascuno di durata l(t), m il numero di processori P . D un tempo massimo.
Si determini, se esiste, un assegnamento dei task ai processori che rispetti il tempo massimo, cioè una funzione A : T $\rightarrow$ IN (dai task al momento di inizio) tale che per ogni u >= 0 il numero dei task t per i quali A(t) <= u < A(t) + l(t) è minore o uguale a m e per ogni t, A(t) + l(t) <= D.
Si risolva il problema utilizzando un algoritmo di ricerca in ampiezza.

**Correzione stringa a stringa (ricerca in ampiezza)**
Date due stringhe x e y, qual è, se esiste, il numero minore di passi per ottenere la stringa y dalla stringa x utilizzando una sequenza di operazioni dei seguenti 2 tipi: cancellazione di simboli; scambio di simboli adiacenti.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Correzione stringa a stringa (ricerca in profondità)**
Date due stringhe x e y, determinare, se esiste, una sequenza di passi per ottenere la stringa y dalla stringa x utilizzando una sequenza di operazioni dei seguenti 2 tipi: cancellazione di simboli; scambio di simboli adiacenti.
Si risolva il problema utilizzando una ricerca in profondità.

**Correzione stringa a stringa (best first)**
Date due stringhe x e y, determinare, se esiste, una sequenza di passi per ottenere la stringa y dalla stringa x utilizzando una sequenza di operazioni dei seguenti 2 tipi: cancellazione di simboli; scambio di simboli adiacenti.
Si risolva il problema utilizzando una ricerca best first.

Come funzione di valutazione si può usare la distanza di Hamming delle 2 stringhe.

**Correzione stringa a stringa (hill climbing)**
Date due stringhe x e y, determinare, se esiste, una sequenza di passi per ottenere la stringa y dalla stringa x utilizzando una sequenza di operazioni dei seguenti 2 tipi: cancellazione di simboli; scambio di simboli adiacenti.
Si risolva il problema utilizzando una ricerca hill climbing.
Come funzione di valutazione si può usare la distanza di Hamming delle 2 stringhe.

**Hitting string (ricerca in profondità)**
Si consideri un insieme finito S di stringhe di lunghezza N sull'alfabeto {0,1,2}.
Determinare, se esiste, una stringa x di lunghezza N sull'alfabeto {0,1} tale che per ogni stringa $s \in S$ d(s,x) >=0, dove d è la distanza di Hamming.
Si risolva il problema utilizzando una ricerca in profondità.

**Hitting string (ricerca in ampiezza)**
Si consideri un insieme finito S di stringhe di lunghezza N sull'alfabeto {0,1,2}.
Determinare, se esiste, una stringa x di lunghezza N sull'alfabeto {0,1} tale che per ogni stringa $s \in S$ d(s,x) >0, dove d è la distanza di Hamming.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Hitting string (best first)**
Si consideri un insieme finito S di stringhe di lunghezza N sull'alfabeto {0,1,2}.
Determinare, se esiste, una stringa x di lunghezza N sull'alfabeto {0,1} tale che per ogni stringa $s \in S$ d(s,x) >0, dove d è la distanza di Hamming.
Si risolva il problema utilizzando una ricerca best first.
Come funzione di valutazione si può il numero stringhe con distanza >0 dalla soluzione parziale.

**Hitting string (hill climbing)**
Si consideri un insieme finito S di stringhe di lunghezza N sull'alfabeto {0,1,2}.
Determinare, se esiste, una stringa x di lunghezza N sull'alfabeto {0,1} tale che per ogni stringa $s \in S$ d(s,x) >0, dove d è la distanza di Hamming.
Si risolva il problema utilizzando una ricerca hill climbing.
Come funzione di valutazione si può il numero stringhe con distanza >0 dalla soluzione parziale.

**Massima sottosequenza comune (ricerca in profondità)**
Si consideri un insieme finito S di stringhe ed un intero K.
Determinare, se esiste, una stringa x di lunghezza maggiore o uguale a K che sia sottosequenza di ogni stringa $s \in S$.
Si risolva il problema utilizzando una ricerca in profondità.

**Massima sottosequenza comune (ricerca in ampiezza)**
Si consideri un insieme finito S di stringhe ed un intero K.
Determinare, se esiste, una stringa x di lunghezza maggiore o uguale a K che sia sottosequenza di ogni stringa $s \in S$.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Massima sottosequenza comune (ricerca branch and bound)**
Si consideri un insieme finito S di stringhe ed un intero K.
Determinare, se esiste, la stringa x di lunghezza maggiore che sia sottosequenza di ogni stringa $s \in S$.
Si risolva il problema utilizzando una ricerca branch and bound.
Come funzione di valutazione si usi la lunghezza della soluzione parziale.

**Set packing (ricerca in profondità)**
Si consideri una collezione finita C di insiemi finiti ed un intero K.
Determinare, se esiste, una sottocollezione di almeno K insiemi mutuamente disgiunti di C.
Si risolva il problema utilizzando una ricerca in profondità.

**Set packing (ricerca in ampiezza)**
Si consideri una collezione finita C di insiemi finiti ed un intero K.
Determinare, se esiste, una sottocollezione di almeno K insiemi mutuamente disgiunti di C.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Matching 3-dimensionale (ricerca in profondità)**
Si consideri un insieme $M \subseteq W \times X \times Y$, dove W,X,Y sono insiemi disgiunti di q elementi.
Determinare, se esiste, un sottoinsieme M' di M tale che $|M'| = q$ e per ogni m1, m2 $\in M'$ d(m1,m2) =3, dove d è la distanza di Hamming.
Si risolva il problema utilizzando una ricerca in profondità.

**Matching 3-dimensionale (ricerca in ampiezza)**
Si consideri un insieme $M \subseteq W \times X \times Y$, dove W,X,Y sono insiemi disgiunti di q elementi.
Determinare, se esiste, un sottoinsieme M' di M tale che $|M'| = q$ e per ogni m1, m2 $\in M'$ d(m1,m2) =3, dove d è la distanza di Hamming.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Minimo tempo di trasmissione (ricerca in profondità)**
Siano G=(V,E) un grafo, $V_0$ un sottoinsieme dei vertici ed un intero K.

Determinare se un messaggio può essere trasmesso in tempo K da $V_0$ a tutto il grafo cioè se esiste una sequenza $V_0, E_1, V_1, E_2, \ldots, E_k, V_k$ tale che $V_i \subseteq V$, $E_i \subseteq E$, $V_k = V$ e che:

1) ogni arco in $E_i$ ha un estremo in $V_{i-1}$ 2)
2) due archi in Ei non hanno un estremo in comune
3) 3) $V_i = V_{i-1} \cup \{v : \{u,v\} \in E_i\}$

Si risolva il problema utilizzando una ricerca in profondità.

**Minimo tempo di trasmissione (ricerca in ampiezza)**

Siano G=(V,E) un grafo, $V_0$ un sottoinsieme dei vertici ed un intero K.
Determinare se un messaggio può essere trasmesso in tempo K da $V_0$ a tutto il grafo cioè se esiste una sequenza $V_0, E_1, V_1, E_2, \ldots, E_k, V_k$ tale che $V_i \subseteq V$, $E_i \subseteq E$, $V_k = V$ e che:

1) ogni arco in $E_i$ ha un estremo in $V_{i-1}$ 2)
2) due archi in Ei non hanno un estremo in comune
3) $V_i = V_{i-1} \cup \{v : \{u,v\} \in E_i\}$

Si risolva il problema utilizzando una ricerca in ampiezza.

**Cammino lungo (ricerca in profondità)**

Sia G=(V,E) un grafo, in cui ogni arco ha associata una lunghezza e sia K un intero.
Dati due vertici determinare, se esiste un cammino di lunghezza almeno K.
Si risolva il problema utilizzando una ricerca in profondità.

**Cammino lungo (ricerca in ampiezza)**

Sia G=(V,E) un grafo, in cui ogni arco ha associata una lunghezza e sia K un intero.
Dati due vertici determinare, se esiste un cammino di lunghezza almeno K.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Problema del postino di campagna (ricerca in profondità)**

Siano G=(V,E) un grafo, in cui ogni arco ha associata una lunghezza, un sottoinsieme E' degli archi ed un intero K.
Dato un vertice V0 determinare, se esiste, un circuito (cammino chiuso da V0 a V0) che include tutti gli archi di E' ed ha lunghezza inferiore a K.
Si risolva il problema utilizzando una ricerca in profondità.

**Problema del postino di campagna (ricerca in ampiezza)**

Siano G=(V,E) un grafo in cui ogni arco ha associata una lunghezza, un sottoinsieme E' degli archi ed un intero K.
Dato un vertice V0 determinare, se esiste, un circuito (cammino chiuso da V0 a V0) che include tutti gli archi di E' ed ha lunghezza inferiore a K.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Problema del postino di campagna (best first)**
Siano G=(V,E) un grafo in cui ogni arco ha associata una lunghezza, un sottoinsieme E' degli archi ed un intero K.
Dato un vertice V0 determinare, se esiste, un circuito (cammino chiuso da V0 a V0) che include tutti gli archi di E' ed ha lunghezza inferiore a K.
Si risolva il problema utilizzando una ricerca best first.
Come funzione di valutazione si può usare il numero di archi di E' presenti nel circuito.

**Problema del postino di campagna (hill climbing)**
Siano G=(V,E) un grafo in cui ogni arco ha associata una lunghezza, un sottoinsieme E' degli archi ed un intero K.
Dato un vertice V0 determinare, se esiste, un circuito (cammino chiuso da V0 a V0) che include tutti gli archi di E' ed ha lunghezza inferiore a K.
Si risolva il problema utilizzando una ricerca hill climbing.
Come funzione di valutazione si può usare il numero di archi di E' presenti nel circuito.

**Problema del postino di campagna (branch and bound)**
Sia G=(V,E) un grafo in cui ogni arco ha associata una lunghezza, un sottoinsieme E' degli archi ed un intero K.
Dato un vertice V0 determinare, se esiste, un circuito (cammino chiuso da V0 a V0) che include tutti gli archi di E' di lunghezza minima.
Si risolva il problema utilizzando una ricerca branch and bound.
Come funzione di valutazione usare la lunghezza della soluzione parziale.

**Bandwidth (ricerca in profondità)**
Siano G=(V,E) un grafo e un intero $K \leq |V|$.
Determinare, se esiste, una funzione iniettiva $f: V \rightarrow \{1,2,\ldots, |V|\}$ tale che per ogni arco $\{u,v\} \in E$ $|f(u)-f(v)| < K$.
Si risolva il problema utilizzando una ricerca in profondità.

**Bandwidth (ricerca in ampiezza)**
Siano G=(V,E) un grafo e un intero $K \leq |V|$.
Determinare, se esiste, una funzione iniettiva $f: V \rightarrow \{1,2,\ldots, |V|\}$ tale che per ogni arco $\{u,v\} \in E$ $|f(u)-f(v)| < K$.
Si risolva il problema utilizzando una ricerca in ampiezza.

**Giochi: da risolvere con un algoritmo di minimax**

**Soluzione gioco filetto**
Si scriva un programma che, data una configurazione del gioco del filetto, determini la mossa che porta alla situazione più favorevole per il giocatore (possa portare alla vittoria se possibile, altrimenti possa portare al pareggio, altrimenti sia una mossa qualsiasi)
Si costruisca l'albero delle varianti e si determini la mossa applicando un algoritmo di minimax

**Soluzione gioco Kayles generalizzato**
Dato un grafo G si consideri il seguente gioco:
I due giocatori rimuovono alternativamente un vertice e tutti i vertici adiacenti. Perde chi rimane senza vertici da rimuovere.
Esiste una strategia vincente per il primo giocatore?
Si costruisca l'albero delle varianti e si determini la mossa applicando un algoritmo di minimax

**Soluzione gioco Hex generalizzato**
Dato un grafo G=(V,E) e due vertici iniziali s, t si consideri il seguente gioco:
I due giocatori (Bianco e Nero) scelgono alternativamente un vertice in V-{s,t} e lo colorano del proprio colore. Bianco vince se e solo se riesce a realizzare un cammino da s a t di vertici bianchi.
Esiste una strategia vincente per il giocatore Bianco?
Si costruisca l'albero delle varianti e si determini la mossa applicando un algoritmo di minimax

**Progetti per due persone**

**Derivatore simbolico**
Si scriva un programma che data una espressione (espressione parentesizzata che può convolgere operatori aritmetici, elevamento a potenza, funzioni trigonometriche, funzione logaritmo e composizione di funzioni) in una variabile ne restituisca la derivata simbolica.

**Interprete linguaggio**
Si scriva un semplice interprete per un linguaggio L a scelta (eventualmente semplificato). Il programma deve porter eseguire un programma sorgente scritto nel linguaggio.
Suggerimento: seguire Lisp di Winston, Horn, 68-xx-1989-8 Cap. 18

**Pianificatore automatico per mondo blocchi**

Realizzare un semplice pianificatore per il mondo dei blocchi che lavori con ricerca in profondità o in ampiezza. Per i dettagli vedere:
Lisp di Winston, Horn, 68-xx-1989-8 Cap. 21

**Traduttore automatico**
Realizzare un semplice traduttore che dati un dizionario ed alcune regole grammaticali di base possa tradurre un testo inglese in italiano.
Suggerimento: considerare Lisp di Winston, Horn, 68-xx-1989-8 Cap. 28-29