**CS156 Practice Midterm 2014**

1. **Briefly explain what the Turing Test is. Also define the term agent in the context of AI.**

Group: Jannette Pham-Le Nikita Pankratov Daniel Kenrick

1. **The Turing test** is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human.
   1. A computer passes the test if a human interrogator, after posing questions to the computer and seeing the responses, cannot distinguish the responses from those of a human. Such a computer would need to be able to:
      1. *Do* ***natural language processing*** *to enable it to communicate successfully in English*
      2. *Use some kind of* ***knowledge representation*** *to store what it hear.*
      3. *Be able to do* ***automated reasoning*** *to use the stored information to answer questions and draw conclusions.*
      4. *Do* ***machine learning*** *to adapt to new circumstances and to detect and extrapolate patterns.*

- a human communicates via an I/O device with an entity known as "person"  
- human asks any question to the "person" and reads the response  
- After a certain interaction the human is asked if he deems the "person" to be an A.I. or another human being  
- The test passes if the human deems the "person" a human being when in fact the "person" is an A.I.  
- The test fails otherwise.

1. An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuator. For example:
   1. *A human agent has eyes, ears, and other organs for sensors and hands, legs and vocal tracts, etc for actuators*
   2. *A robotic agent might have cameras and infrared range finders for its sensors, and various motors for actuators*
   3. *A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on its environment by displaying stuff on the screen.*
2. **What is the PEAS description of a task environment? Give a fully spelt-out example of an episodic task environment.**

Group: Steve Wang, Dmitri Dimov, Andrew Vinh

1. PEAS stands for:

[P]erformance - How efficiently the agent reaches it's goal.

[E]nvironment - The space where agent is located and receives its percepts.

[A]ctuators - All features that can manipulate the surrounding environment.

[S]ensors - All agent features that interpret surrounding environment.

1. An example of PEAS would be a quality control robot.
   1. Performance - fast, accurate
   2. Environment - bottle
   3. Actuators - control arm, conveyor belt
   4. Sensors - camera, flame

**Properties of Task Environments:**

1. Fully observable vs partially observable
   1. chess vs taxi
2. Single agent vs multiagent
   1. vacuum cleaner vs chess(competitive)/ taxi(co-operative)
3. Deterministic vs stochastic
   1. next state determined by prev state vs stochastic
4. Episodic vs sequential
   1. QA robot checking for defective bottles vs chess
5. Static vs dynamic
   1. non-changing environment(chess) vs changing environment(taxi)
6. Discrete versus continuous
7. Known versus unknown
8. **Briefly describe iterative deepening depth first search and analyze its time complexity.**

Group: Andres Chorro, Justin Tieu, Gabe Guevara

**Depth-first search (DFS)** always expands the deepest node in the current frontier of the search tree. It goes until it finds a solution, or a node with no successors, then moves to the next-deepest branch, and continues.

**Depth-limited Search** runs the same way, but with a predetermined limit, L. This means that nodes at depth L are treated like they have no successors.

**Iterative deepening depth-first search (IDS)** performs Depth-limited Search with L = 0. If it fails, it searches AGAIN with L = 1. It keeps searching with depth of L+1 until it finds a solution or fails.

**Time complexity of IDS:**

The nodes at the bottom level (depth d) are generated once. The nodes on the level above that are generated twice, and so on, up to the children of the root, which is generated d times. So the total number of nodes generated in the worst case (where b is the branching factor) is: N(IDS) = (d)b + (d-1)b^2 + ... + (1)b^d.

Which gives a time complexity O(b^d)

1. **Give a concrete example where a problem solving agent using *A*⋆-search would not traverse a graph in the same way as one using breadth-first search. Give an example where depth first search outperforms *A*⋆-search.**

S  
 c=1 c=2 c=3  
 A B C  
 c=3  
 G

S is start, G is goal, heuristic is euclidean. and nodes S,A,B,C,G are in euclidean space, c is cost

1. A\* Search would check node B first given it's lower actual + heuristic value, then after finding no connecting nodes it would back up and check node A, because it's actual + heuristic cost is less than node C. It would then Find the goal. Breadth-first-search would first check all the nodes on the first level finding the goal after looking at Node C. in this way it is different than A\*.
2. Depth-first-search would start at the past lowest initial cost, checking node A, then finding the goal, out performing A\*
3. **Write a short python program which takes its command line arguments and sums them together. This program should make use of at least one function definition.**

Group: David Nguyen, Christopher Raleigh, and Huzefa Siyamwala

import sys

def sum\_list():

sum = 0  
 for index in range(1, len(sys.argv)):  
 sum += int(sys.argv[index])  
 return sum

print "The sum is: " + str(sum\_list())

1. **Given two admissible *A*⋆ heuristic explain how to make a new heuristic which performs at least as well as either of them.**

Group: Katharine Brinker, Edward Ciotic

An A\* heuristic is admissible if it never overestimates the cost to a solution.

Given two admissible heuristics h1 and h2 at node n, it can be assumed that h1 will sometimes perform better than h2, and h2 sometimes perform better than h1. We can therefore make a new heuristic h at node n by taking h= max{h1(n), h2(n)}. This always picks the best-performing heuristic for a given situation and therefore performs at least as well as h1 or h2. h is guaranteed to be admissible because it's running an admissible heuristic.

1. **Briefly explain how genetic programming and local beam search are related hill-climbing algorithms.**

Group: David Smith & Jake Karnes

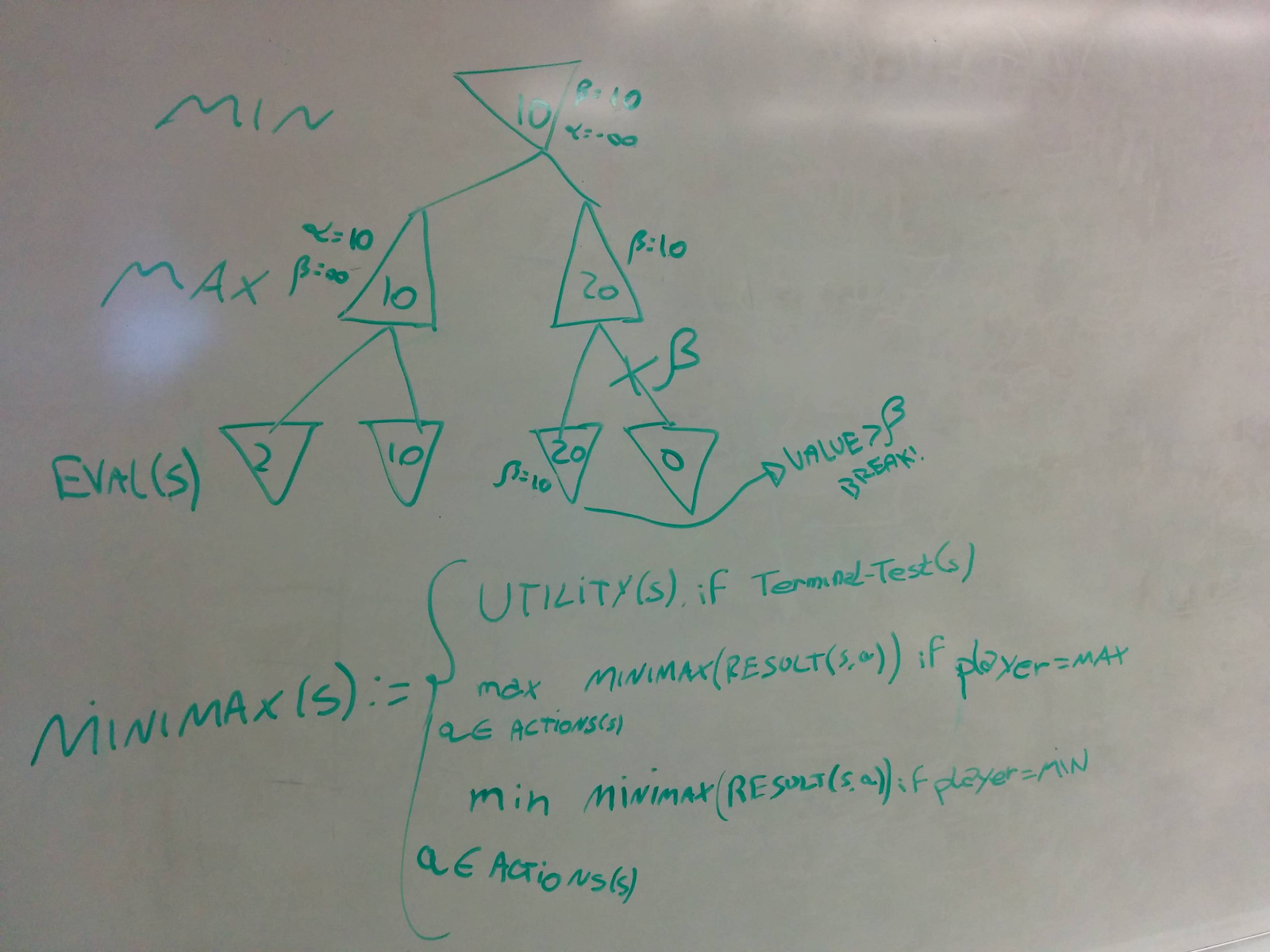
In Hill Climbing, we move from the current state to the next best successor state, if one exists. This is in pursuit of finding the best possible solution to the problem.

Local Beam Search and Genetic Algorithms are similar in that they both track K current states. They differ in how they generate successor states. Local Beam Search generates all successors of current states and picks the K best to be the next set considered. Genetic Algorithms generates K successors by stochastically combining the best pairs from the current set.

1. **What is the minimax function? Given an example of a situation in which a beta-cut might arise while running the minimax algorithm with alpha-beta pruning.**

Group: Bruno Lopes, Shubhangi Rakhonde, Alejandro Jimenez

The minimax function is used in adversarial search to find a value for an optimal move in a given state.

Formal definition and beta-pruning example in the image:

In the beta-pruning example, MIN has a backed-up value of 10, and starts exploring its second child. When it reaches the node with the value ' 20' , it can cut this branch since MAX will always take ' 20' or something better, and MIN already has a backed-up value of ' 10' , so it would never choose something bigger than 20.

1. **Give pseudocode for the AC3 algorithm.**

---------------------

1. **Consider the following situation for four light switches on the control panel of a nuclear power plant: (a) The first and last can never both all on. (b) At least one light must be on. Express this as a propositional logic knowledge base.**

Group:David Schechter, Zayd Hammoudeh, Geetika Garg

Solution:

L1 - propositional symbol for if First Light is On. True if on, False otherwise

L2 - propositional symbol for if Second Light is On. True if on, False otherwise

L3 - propositional symbol for if Third Light is On. True if on, False otherwise

L4 - propositional symbol for if Fourth (Last) Light is On. True if on, False otherwise

R1: NOT (L1 AND L4)

R2: L1 OR L2 OR L3 OR L4

R1 AND R2

[NOT (L1 AND L4)] AND [L1 OR L2 OR L3 OR L4]

**CS156 Practice Midterm 2012**

1. **Briefly say what the Total Turing Test is, what a rational agent is.**

Besides the requirements of Turing Test, the interrogator can also test the perceptual abilities of the subject (requiring [computer vision](http://en.wikipedia.org/wiki/Computer_vision)) and the subject's ability to manipulate objects.

A rational agent always pick the option amongst all the possibilities where he expects to obtain maximum profit.

1. **Give the formal way to specify a problem for a problem solving agent.**
2. **Give an example problem and then explain how iterative deepening search might search the environment of this problem to find a goal. Explain the run time and space complexity of IDS.**
3. **What is the difference between Greedy-Best-First Search and A\* search? Give an example situation where the latter might perform better than the former.**
4. **Give an example of each of the following programming language feature in Python: generators, coroutines, lambda.**
5. **Describe each of the following local search algorithms: (a) random-hill climbing with restarts, (b) local beam search.**
6. **What is an adversarial search problem? Zero-sum game? Optimal Strategy?**
7. **Give the minimax algorithm. Give an example of a situation in which an alpha-cut might be made. Give an example of a situation in which an beta-cut might be made.**
8. **Briefly explain the AC-3 algorithm, and show how it might be applied to a particular CSP.**
9. **Give a resolution refutation of the following clauses: {a, not b}, {b, not c}, {c}, {not a}.**