Chapter 2: Intelligent Agents

(2.1): Agents and Environments

(2.2): The Concept of Rationality

(2.3): The Nature of Environments Be comfortable with the PEAS description and environment properties.

(2.4): The Structure of Agents You may be asked to pick the best agent type for some problem and justify your answer.

(2.5): Summary Go through the chapter summary.

Chapter 3: Solving Problems by Search

(3.1): Problem-Solving Agents Be comfortable defining a search problem.

(3.2): Example Problems

(3.3): Search Algorithms & Uniform Search Strategies Ignore sections 3.4.4 and 3.4.5 for the exam.

(3.5): Informed (Heuristic) Search Strategies You may be asked to solve a search problem by hand.

(3.6): Heuristic Functions

(3.7): Summary Go through the chapter summary. FOCUS ON A* algorithm

Chapter 4: Search in Complex Environments

(4.1): Local Search and Optimization Problems

Hill-climbing search

Algorithm 0.1 Hill-climbing search

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← problem.INITIAL
    while true do
        neighbor ← a highest-valued successor state of current
        if VALUE(neighbor) ≤ VALUE(current) then return current
        end if
        current ← neighbor
    end while
end function
```

Simulated Annealing

Algorithm 0.2 Simulated Annealing

```
function SIMULATED-ANNEALING(problem, Schedule) returns a solution state
    current ← problem.INITIAL
    for t = 1 to ∞ do
        T ← SCHEDULE(t)
        if T == 0 then return current
        end if
        next ← a randomly selected successor of current
        ΔE ← VALUE(current) − VALUE(next)
        if ΔE > 0 then current ← next
        else current ← next only with probability e^(ΔE/T)
        end if
    end for
end function
```

Evolutionary algorithms

Algorithm 0.3 Genetic Algorithm Pseudocode

```
function GENETIC-ALGORITHM(population, fitness) returns an individual
    repeat
        weights ← WEIGHTED-BY(population, fitness)
        population2 ← empty list
        for i = 1 to SIZE(population) do
            parent1, parent2 ← WEIGHTED-RANDOM-CHOICES(populati
            child ← REPRODUCE(parent1, parent2)
            if small random probability then child ← MUTATE(child)
            end if
            add child to population2
        end for
        population ← population2
    until some individual is fit enough, or enough time has elapsed
    return the best individual in population, according to fitness
end function
function REPRODUCE(parent1, parent2) returns an individual
    n ← LENGTH(parent1)
    c ← random number from 1 to n
    return Append(SUBSTRING(parent1, 1, c), SUBSTRING(parent2, c+1, n))
end function
```

. . . and everything related to Evolutionary algorithms that I covered IGNORE TABU SEARCH in class (especially: EVERYTHING about GENETIC ALGORITHM)

Chapter 5: Adversarial Search and Games

(5.1): Game Theory

(5.2): Optimal Decision in Games You may be asked to solve an adversarial problem by hand using Min-Max and alpha-beta pruning. Ignore section 5.2.2.

(5.3): Summary Go through the chapter summary.

Chapter 6: Constraint Satisfaction Problems

(6.1): Defining CSPs You may be asked to formally define a constraint satisfaction problem.

(6.2): Constraint Propagation: Inference in CSPs Ignore sections 6.2.4 and 6.2.5.

(6.3): Backtracking Search for CSPs Ignore sections 6.3.3 and 6.3.4.

(6.4): Summary Go through the chapter summary.

Chapter 7: Evolutionary Algorithms

Chapter 8: Ant Colony Optimization