INF5390 oblig 1

mathiajj and martinvl

Problem 1

a) **False**

Acting rationally means maximizing performance based on what the agent knows (or should know). If the agant is unable to achieve maximum performance due to insufficient information this is clearly not the case.

So even though the agent's performance might be limited by insufficient information, it might perfectly well be rational.

b) **True**

In a scenario where the agent should output either `true` or `false` every time it receives input. The agent scores points if it outputs `true` when it has received an even amount of inputs, and `false` otherwise. The goal is to score as many points as possible.

No pure reflex agent can act rational in this environment because it has to take into account the *state* of the environment.

c) **True**

For instance environments with no room for choices to be made, or where all choices yield the same performance.

d) **False**

The input to the agent program is the set of perceptions. The agent program might for instance also contain an environment model, which it will include as input to the agent function.

e) **True**

See c)

f) **False** Rationality might also depend on the built-in knowledge of the agent.

g) **False**

Assuming the agent starts with a finite amount of money there exists an environment such that the agent always gets the worst cards. If the agent plays itself in this environment, it will either lose all it's money on blinds or on lost hands.

Problem 2

a)

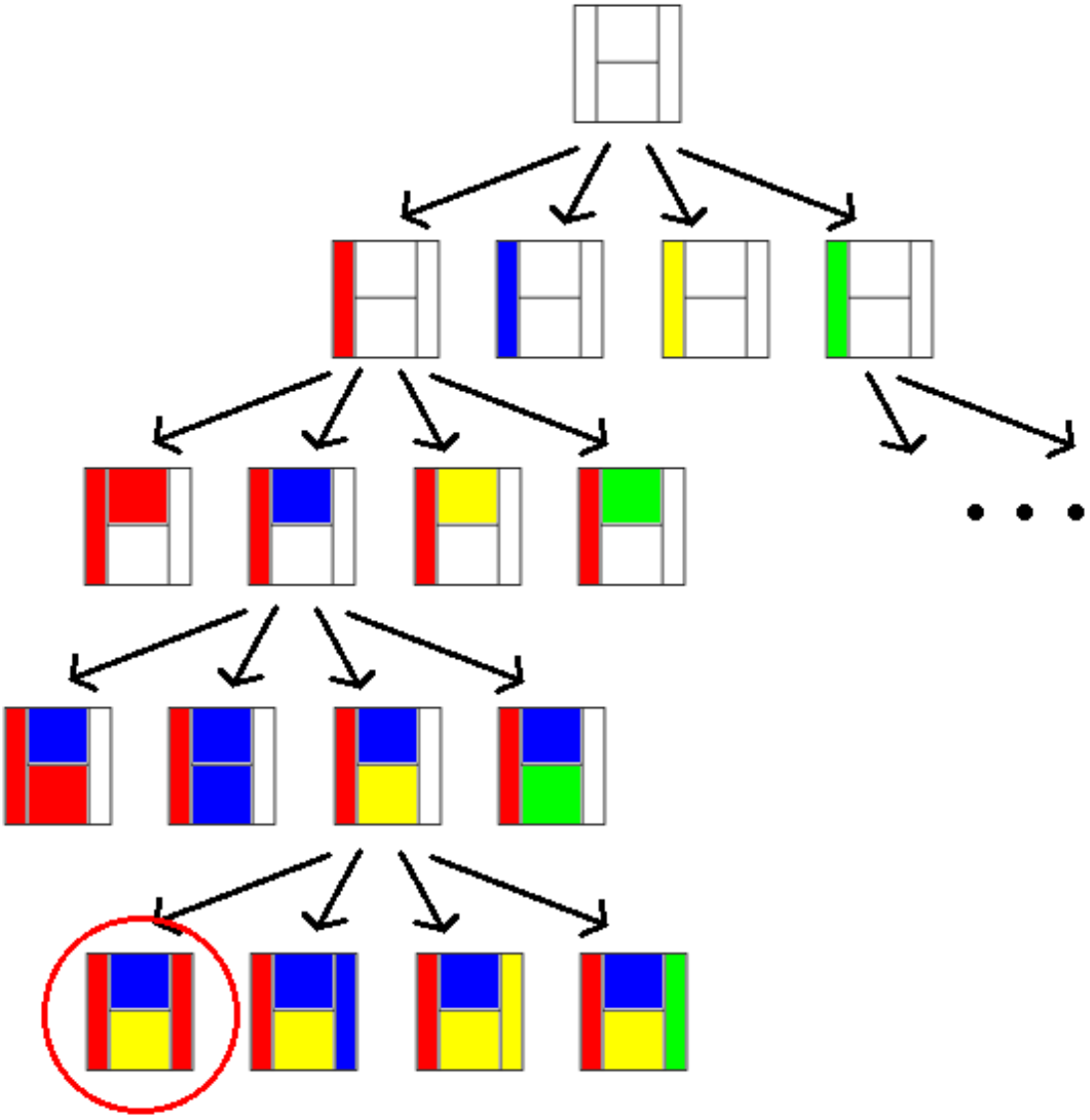
Number the regions 1, ..., n, and define a function *f* that given a partial coloring of the map returns whether a specific region can have a specific color. This function is worst case linear in time, but by four color theorem it is constant on average.

Define a root node in a tree as the state where the colors of all regions is undetermined. Then add 4 nodes connected to the root, each representing a coloring of region 1. Repeat this process for each new node.

The result will be a tree where level *i* represents the coloring of region *i*, and the path to a leaf node represents the coloring of the entire map.

Search the tree for a path from the root to any leaf node where *f* is true for all nodes in the path. In order to find the lowest possible number of colors all such paths must be expanded.

b)



c)

DFS. Same time complexity as BFS (since the solution is a leaf node), but much lower memory footprint. Uniform cost, depth limited and bi-directional search don't really apply in this situation.

DFS is likely to find a solution faster than iterative deepening. When a solution is found, the branching factor can be reduced in the unexplored part of the tree. For instance, if a solution using four colors is found, there is no point in searching for solutions with more colors than three.

d)

The time complexity of this algorithm is $O(4^n)$, which is worse than the best known solutions $O(2^n n)$ time. Memory usage is O(n) (there is no need to actually store it as a graph), which is good.