WOA7001 Post-test

Name:YangZhenyu

Matric number:s2118292

In order to better understand our project, I have made some understanding of optimization algorithms in pre-test and pro-test respectively.

Here is the post-test: An understanding of dynamic programming optimization algorithms.

Backtracking is a class of algorithms used to find solutions to certain computational problems, particularly constraint satisfaction problems, that progressively build candidates for solutions and immediately abandon candidates (" backtracking ") valid solutions when it is determined that the candidates cannot be completed.

The classic textbook example of using backtracking is the Eight Queens puzzle, which requires all permutations of eight chess queens on a standard board so that no queen attacks another queen. In the common backtracking method, a partial candidate is a permutation of k queens in k rows before the board, all in different rows and columns. Any partial solution that involves two queens attacking each other can be abandoned.

Backtracking should only be applied to problems that acknowledge the concept of "partial solution candidates" and whether a relatively quick test of an effective solution can be accomplished. For example, it is useless for locating a given value in an unordered table. However, when it applies, backtracking is generally much faster than a strong enumeration of all complete candidates, because it can eliminate many candidates in a single test.

Backtracking is an important tool for solving constraint satisfaction problems, such as crossword puzzles, verbal arithmetic, Sudoku, and many other puzzles. It is often the most convenient technique to parse for knapsack problems and other combinatorial optimization problems. It is also the basis for so-called logical programming languages such as Icon, Planner, and Prolog.

Backtracking relies on a user-given "black box process" that defines the problem to be solved, the nature of partial candidates, and how they extend to full candidates. As such, it is a meta-heuristic algorithm rather than a specific algorithm - although unlike many other meta-heuristics, it can guarantee finding all solutions to a finite problem in a finite amount of time.