

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2018-2019** |
| **Major**: | Software Engineering |
|  |  |
|  |  |
|  |  |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Backtracking Framework | | | |
| Date | | Nov 16，2018 | Type | | □Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements**    1. Review the design and implementation principles of backtracking algorithm to solve a practical problem    2. Understand the design of backtracking framework using C++, especially the iterator inner class    3. Use the backtracking framework to solve a specific problem | | | | | |
| 1. **Experimental environment (**platform and software**)**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results)   Task 1   * 1. A chessboard has eight rows and eight columns. From its current position, a knight’s next position will be either two rows and one column or one row and two columns from current position. Develop and validate a program to show the moves of a knight traversing a chessboard. Your program should      + Ask the use to input the start position      + Based on the backtrack framework sent to you   **Procedure:**   1. Revise the application.cpp. The first method is initialize(). It is obvious that we need to initialize the chess board like putting 0s into every position. The second one is getStartPosition(). We need to accept a position to start the traversal of knights and return this Position(x,y). It is also important to assign the sequence number to the start position which I set an extern variable “n” and is 1 at the beginning. 2. In isValid() method, we need to check if the current position is valid. There are two conditions: the position is in the board and there is no knight in this position. As a result, as we had put an sequential number in every position we had traversed, just check whether board[row][column] is 0 or not. 3. In extend(Position p), just assigning n to position p. Similarly in success(position p) we need to check if the last sequence number is 64. In goBack(), we need to reset the position p as 0 and n should minus 1. 4. In method print(), there is a judge for aligning in good order. If board[i][j] is less than 10 we need to print one more space.   The most important methods are the iterators’.There are 8 directions for every knight which want to traverse the next position. So, in its constructor, the direction number was set as 0 and so did it in another one with a parameter. Moreover, the parameter position p must be assigned to curr of iterator.   1. In getNextPossible(), things became complicated since we need to combine it with the method backtrack().   Firstly, direction plus 1 since the default direction is 0 and the row and column of current position were assigned to two variables for future convenience. Except above two variables, there is also a need of two variables for next position and so there came x and y.  Before switch(directions), n plus 1 as we need to get the next position and assign to it. Then I defined the next position in 8 directions in 8 cases. At last, we only need to return the position(x,y). However, there is an indispensable step to use method isValid() before we return the value. If the position x,y is not valid, n should minus 1 since it had plus before.   1. Last method is tryAllNextPossible(), we only need to check if the directions had finished 8 diections.   Code:  Backtrack.cpp  Application.cpp            Main.cpp    Result: | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   The easiest way is to set directions as numbers and try everyone in each case of switch. So that we don’t need to write lots of codes for every suitation as we need to judge as well as assign values to new position, which is a troublesome work.  The greatest problem I have made is that I did not wrote “break” in every case in switch, which caused a confusing result that “Problem has no solution”.  In main.cpp, I also put time.h into header file to calculate time cost. | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |