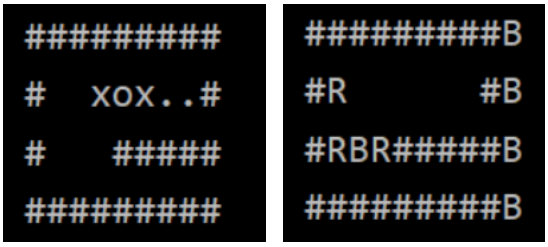
# HW1: Sokoban

ISA 110065504 葉之晴

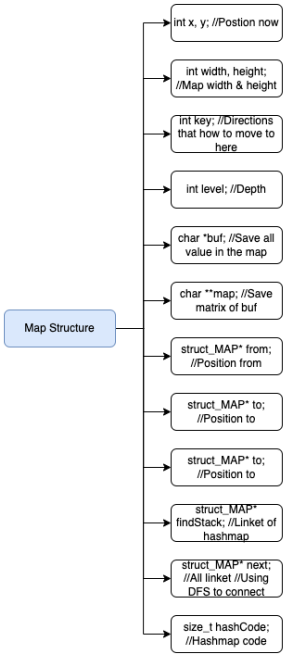
## 1. Briefly describe your implementation.

a. Describe

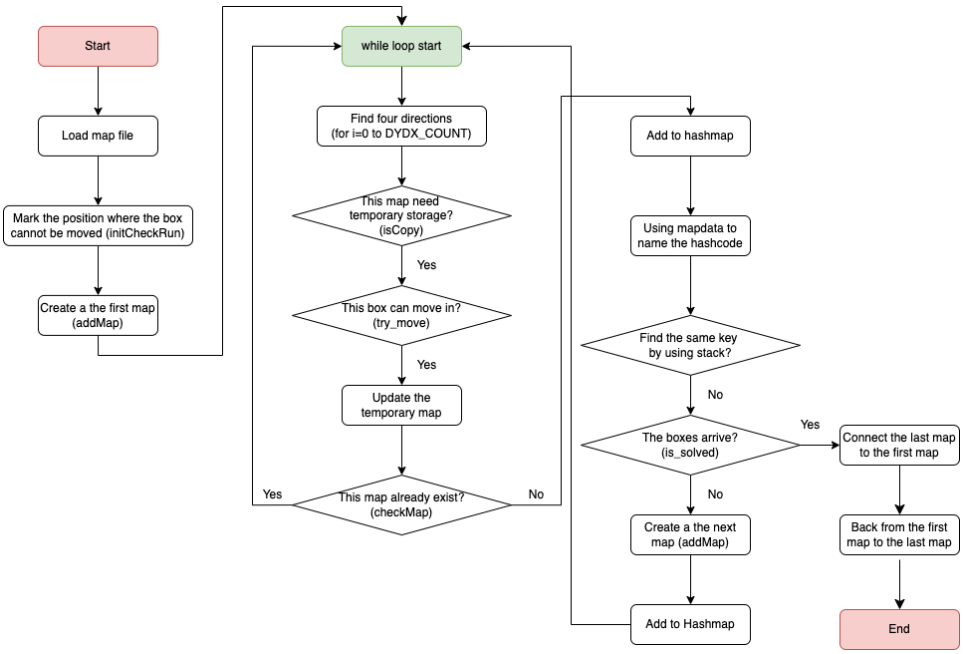
Initially, I mark the position where a box cannot be moved. The example is as the following picture. The picture on the left is the original problem map. For the other, B and R marked in the picture are the positions where the box cannot move in. If it moves in, the game will be dead.




Pthread is written, but the performance is not so good. So, I accelerate the time by using HashMap to find if the map is duplicated. Then, use DFS instead of BFS because of the speed it executed. Moreover, the dead disk has been dealt with when checking the map, which can also reduce the time. The HashMap structure is as the following picture.



b. Flow Chart




2. **What are the difficulties encountered in this homework? How did you solve them?**

   In the beginning, because I used to use online judge before, I was not familiar with this judge method on Linux. Also, because the format of input and output was not clearly defined initially, I spent a little time trying again and again for the platform. I also thank the teaching assistant for answering the question of the format immediately.

   For solving the problem, I programmed the algorithm and made sure that it could be executed first, and then started to do the parallel computing. As a result, it was found that such a program structure was difficult to parallel, so the structure was rewritten as a struct. It does take a lot of time and patience to do this.

   Then, after doing the parallel computing, the time of the program has not been reduced a lot first. I try to analyze the program and found that finding the route took most of the time. So, I began to think about which search algorithm I learned in the data structure course in the past was the fastest, and tested the HashMap to greatly improve the speed. At the same time, it was also found that in the case of pthread, the HashMap speed would be slowed down, so it was not used.

Moreover, to meet the speed-oriented problem-solving strategy, these problems including selecting BFS or DFS, setting the number of buffers in the HashMap, pre-setting which position cannot be moved in, etc., are all been tested, and I chose the method with a shorter time. These make me understand the difficulty of program acceleration.

3. **What are the strengths and weaknesses of pthread and OpenMP?**

The difference between OpenMP and pthread is mainly in the way of compilation. The compilation of OpenMP needs to add the compiler preprocessing instruction #pragma, and the following work such as creating threads needs the compiler to complete. And pthread is a library, all parallel thread creation needs to be done by ourselves, which is a little more troublesome than OpenMP.

In detail, it is no need to specify the number and just add code where there is a loop for OpenMP. It has better applicability because it doesn't limit the software to a pre-set number of threads, but it is more difficult to find the error. On the other hand, for pthread, it creates a bunch of threads at program startup and distributes work to them. However, this approach requires considerable thread-specific code and is not guaranteed to extend reasonably with the number of available processors.