|  |  |  |  |
| --- | --- | --- | --- |
| Easy Board: | Medium Board | Hard Board | Evil Board |
| [3, 1, 9, 7, 0, 5, 2, 0, 0]  [0, 5, 4, 0, 0, 0, 0, 0, 3]  [0, 8, 0, 0, 9, 0, 0, 0, 5]  [0, 7, 0, 6, 5, 3, 0, 0, 0]  [4, 0, 0, 0, 1, 0, 0, 0, 7]  [0, 0, 0, 8, 4, 7, 0, 6, 0]  [9, 0, 0, 0, 8, 0, 0, 3, 0]  [7, 0, 0, 0, 0, 0, 8, 2, 0]  [0, 0, 8, 2, 0, 9, 4, 5, 6] | [0, 0, 0, 6, 2, 0, 0, 4, 3]  [0, 0, 0, 1, 0, 0, 6, 7, 0]  [0, 0, 0, 0, 3, 4, 0, 9, 0]  [0, 3, 5, 0, 0, 8, 0, 0, 0]  [0, 0, 0, 0, 7, 0, 0, 0, 0]  [0, 0, 0, 4, 0, 0, 3, 1, 0]  [0, 1, 0, 8, 4, 0, 0, 0, 0]  [0, 4, 7, 0, 0, 3, 0, 0, 0]  [3, 2, 0, 0, 5, 6, 0, 0, 0] | [0, 9, 7, 0, 0, 0, 0, 0, 0]  [3, 0, 0, 9, 0, 0, 2, 0, 0]  [0, 5, 0, 2, 0, 1, 0, 0, 4]  [0, 0, 0, 0, 0, 3, 8, 0, 6]  [1, 0, 0, 0, 0, 0, 0, 0, 3]  [7, 0, 4, 6, 0, 0, 0, 0, 0]  [8, 0, 0, 3, 0, 5, 0, 9, 0]  [0, 0, 9, 0, 0, 6, 0, 0, 8]  [0, 0, 0, 0, 0, 0, 5, 6, 0] | [0, 8, 2, 0, 6, 0, 0, 0, 0]  [5, 0, 0, 0, 0, 3, 0, 0, 9]  [0, 3, 0, 0, 1, 0, 0, 0, 7]  [8, 6, 7, 0, 0, 0, 0, 0, 0]  [9, 0, 0, 0, 0, 0, 0, 0, 2]  [0, 0, 0, 0, 0, 0, 7, 8, 6]  [7, 0, 0, 0, 8, 0, 0, 2, 0]  [3, 0, 0, 2, 0, 0, 0, 0, 5]  [0, 0, 0, 0, 5, 0, 3, 1, 0] |

Different board settings got from <http://www.websodoku.com>

Backtrack Naïve

For Easy Board : 0.005775217000000055

For Medium Board : 1.8519752600000174

For Hard Board : 0.04015511899979174

For Evil Board : 0.24649119099998984

Backtrack using Minimum Remaining Value:

For Easy Board : 0.02521942699999613

For Medium Board : 0.04343080700000712

For Hard Board : 0.05334200099969166

For Evil Board : 0.10358925500000282

It seems that for some settings the Naïve backtracking works good. As it puts 1 to 9 to each square and carries forward to another square. When most of the number placement works smoothly then it takes little time to complete the puzzle. But if you take a top level view, naïve backtracking took more than 2 seconds to solve the 4 types of puzzle.

On the other hand, I used Minimum Remaining Value[**MRV**] Technique as smart Backtracking. The performance spike is noticeable. In this technique, for each square I could minimize the number of choices based on the constrains so it was faster than naïve technique. Even in the medium board settings, naïve backtrack took 1.8s, where RVT took only 0.04s. On the top level view, all four settings cumulatively took only around 0.2s. Which is *10 times* faster than naïve backtracking.