CMPS 101 - Programming assignment 3 - W13
Solving the Jumping Pegs Problem via Backtracking while using Hashing for MemoizationKeeping
Handed out Tu 2-19-13 Due Sa 3-2-13 in locker

February 28, 2013

1. Design a program for finding a solution to the Jumping Pegs Problem via Backtracking
2. Use Hashing to avoid solving the same board a second time.

Rules of the problem

- | denotes a peg
- *X* denotes a double peg
- A peg | must jump over 2 pegs (over| | or over *X*) and land on a peg
- ```
  |  |  |  |  --->   |  |  X
  ^                  ^

  |  X  |     --->   X  X
  ^                  ^
  ```
- A peg can jump either left or right
- Double pegs can't move and spaces are ignored
- Cant land on an empty spot or double peg.

Input: Start configuration of single and double pegs
Question: Is there a sequence of jumps s.t. all pegs are doubled?
−If not then output "no solution".
−If yes then output a sequence of boards or moves shows how the solution was achieved.
  ```

Begin by:

1. Solving small problems by hand.
   Try to solve the following:

```
| | X | | | | |      --->  no solution
| | | | | |          --->  no solution
| | | | | | | | | |  --->  X X X X X
```

2. Write a recursive backtracking routine for solving the problem

Hashing:

1. Represent the "board" as a sequence of bits: 0 encodes | and 1 encodes X.
   When empty spots appear then you need to left shift.

2. Interpret bit sequences as keys represented in binary that you can hash on.

3. Whenever a board is known to have no solution, then hash it.

4. Before recursing on a new board, check whether it is not in your hash table.
   If it is, then you can skip this recursion.

5. You only need to implement **Insert** and **Search** but not **Delete**.

6. Use open addressing. Keep track of your load factor α. If $\alpha > .\mathbf{2}$, then rehash the whole table content into one **4** times as larger. For the initial hash table size we suggest to use: $m = \mathbf{2^{12}} = 4096$. You are welcome to vary the .**2**, **4**, and $\mathbf{2^{12}}$ constants in sensible ways.

7. Start with the division method and linear probing.

8. Once this works, then use double hashing: $h(k,i) = (h_1(k) + i\,h_2(k)) \mod m$.

   $$\text{frac} := kA \mod 1,\ \ \text{mfrac} := m\,\text{frac},\ \ h_1(k) := \lfloor \text{mfrac} \rfloor,\ \ h_2(k) := \lfloor m(\text{mfrac} \mod 1) \rfloor.$$

9. Finally implement the enlarging your hash table procedure.

10. Input format: A sequence of bits.

Add a short (half page) description of what you observed:
- For what type/size of inputs did the hash table give an advantage?
- If you tuned any parameters, what was your rationale?
- Anything else insightful