# UNIVERSITY OF CALIFORNIA, IRVINE

**CompSci 271: Introduction to Artificial Intelligence**

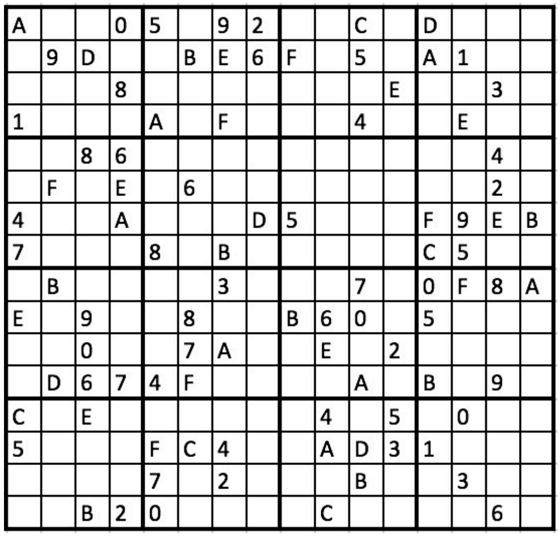
# (Hexadecimal) Sudoku Project report

**Professor: Kalev Kask**

|  |  |
|---|---|
| **Yonglin Chen** | **10244704** |
| **Lintong Luo** | **94681045** |
| **Jiehui Zhang** | **21363170** |
| **Yuanzhe Li** | **14563990** |

## 1. Instruction

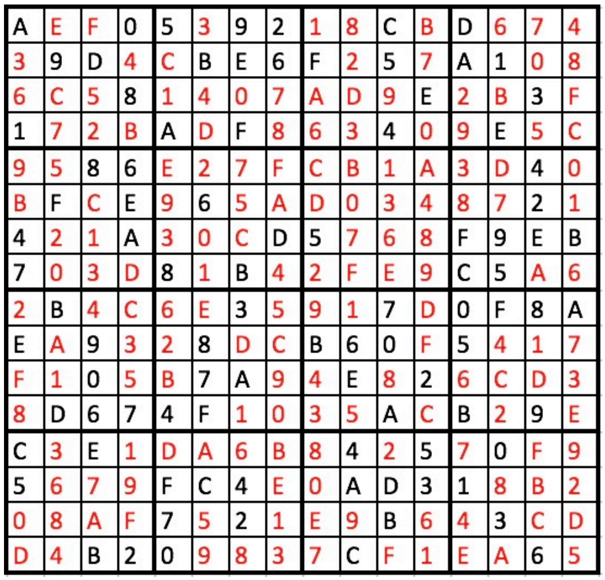
Sudoku is a kind of math problem. It comes from a Japanese phrase which means "the number must be single" and it is an easy logic-based number placement puzzle designed for a single player, also it likes a crossword puzzle. This puzzle is same as the grid of little boxes called "cells". Normally sudoku puzzle is a composition of 81 cells which are divided into nine rows, nine columns and regions. In this project, we use a Hexadecimal Sudoku, with 256 cells, 16*16. The puzzle comes with some of the cells already filled in, as the following:



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | | | 0 | 5 | | 9 | 2 | | | C | | D | | | |
| | 9 | D | | | B | E | 6 | F | | 5 | | A | 1 | | |
| | | | 8 | | | | | | | | E | | | 3 | |
| 1 | | | | A | | F | | | | 4 | | | E | | |
| | | 8 | 6 | | | | | | | | | | | 4 | |
| | F | | E | | 6 | | | | | | | | | 2 | |
| 4 | | | A | | | | D | 5 | | | | F | 9 | E | B |
| 7 | | | | 8 | | B | | | | | | C | 5 | | |
| | B | | | | | 3 | | | | 7 | | 0 | F | 8 | A |
| E | | 9 | | | 8 | | | B | 6 | 0 | | 5 | | | |
| | | 0 | | | 7 | A | | | E | | 2 | | | | |
| | D | 6 | 7 | 4 | F | | | | | A | | B | | 9 | |
| C | | E | | | | | | | 4 | | 5 | | 0 | | |
| 5 | | | | F | C | 4 | | | A | D | 3 | 1 | | | |
| | | | | 7 | | 2 | | | | B | | | 3 | | |
| | | B | 2 | 0 | | | | | C | | | | | 6 | |

Figure 1. Initial State

The solution numbers are marked in red as shown in figure 2:



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | E | F | 0 | 5 | 3 | 9 | 2 | 1 | 8 | C | B | D | 6 | 7 | 4 |
| 3 | 9 | D | 4 | C | B | E | 6 | F | 2 | 5 | 7 | A | 1 | 0 | 8 |
| 6 | C | 5 | 8 | 1 | 4 | 0 | 7 | A | D | 9 | E | 2 | B | 3 | F |
| 1 | 7 | 2 | B | A | D | F | 8 | 6 | 3 | 4 | 0 | 9 | E | 5 | C |
| 9 | 5 | 8 | 6 | E | 2 | 7 | F | C | B | 1 | A | 3 | D | 4 | 0 |
| B | F | C | E | 9 | 6 | 5 | A | D | 0 | 3 | 4 | 8 | 7 | 2 | 1 |
| 4 | 2 | 1 | A | 3 | 0 | C | D | 5 | 7 | 6 | 8 | F | 9 | E | B |
| 7 | 0 | 3 | D | 8 | 1 | B | 4 | 2 | F | E | 9 | C | 5 | A | 6 |
| 2 | B | 4 | C | 6 | E | 3 | 5 | 9 | 1 | 7 | D | 0 | F | 8 | A |
| E | A | 9 | 3 | 2 | 8 | D | C | B | 6 | 0 | F | 5 | 4 | 1 | 7 |
| F | 1 | 0 | 5 | B | 7 | A | 9 | 4 | E | 8 | 2 | 6 | C | D | 3 |
| 8 | D | 6 | 7 | 4 | F | 1 | 0 | 3 | 5 | A | C | B | 2 | 9 | E |
| C | 3 | E | 1 | D | A | 6 | B | 8 | 4 | 2 | 5 | 7 | 0 | F | 9 |
| 5 | 6 | 7 | 9 | F | C | 4 | E | 0 | A | D | 3 | 1 | 8 | B | 2 |
| 0 | 8 | A | F | 7 | 5 | 2 | 1 | E | 9 | B | 6 | 4 | 3 | C | D |
| D | 4 | B | 2 | 0 | 9 | 8 | 3 | 7 | C | F | 1 | E | A | 6 | 5 |

Figure 2. Goal State

## 2. Algorithms

There are many algorithms can be used to solve Sudoku, such as Brute Force, Stochastic Search, Backtracking Search, Backtracking Search with forward checking, Constraint Satisfaction, Minimum Remaining Values (MRV), Least Constraining Value, etc. We used Backtracking Search and Dancing Links to solve the problem. Dancing Links performed better than Backtracking Search with forward checking.

### 2.1 Backtracking Search with forward checking

Backtracking is a general algorithm for finding all solutions to some computational problems, notably constraint satisfaction problems which incrementally builds candidates to the solutions, and abandons each partial candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid solution. forward checking. Whenever a variable X is assigned, the forward-checking process establishes arc consistency for it: for each unassigned variable V that is connected to X by a constraint, delete from V's domain any value that is inconsistent with the value chosen for X. Minimum Remaining Values (MRV) is the idea to choose the variable with fewest legal values. The Pseudocode for Backtracking Search with Forward Checking is shown as figure 3:

```
BT+FC($A, U, D$)
if $A$ is complete then
  return $A$
end if
Remove a variable $X$ from $U$
for all values $x \in D(X)$ do
  if $X = x$ is consistent with $A$ according to the constraints then
    Add $X = x$ to $A$
    $D' \leftarrow D$ (Save the current domains)
    for all $Y \in U$ (i.e., $Y$ an unassigned variable), $Y - - - X$ (i.e., $Y$ a neighbor of $X$ in the
    constrained graph) do
      Remove values for $Y$ from $D'(Y)$ that are inconsistent with $A$
    end for
    if for all $Y \in U, Y - - - X$, we have $D'(Y)$ not empty then
      $result \leftarrow$ BT+FC($A, U, D'$)
      if $result \neq failure$ then
        return $result$
      end if
    end if
    Remove $X = x$ from $A$
  end if
end for
return $failure$
```

Figure 3. Pseudocode for Backtracking Search with Forward Checking

## 2.2 Dancing Links

Dancing Links is an algorithm by Knuth[2] to solve exact cover problems (also called Algorithm X). An exact cover problem is as follows: given a binary matrix, select a subset *S* of the rows such that each column has exactly one 1 when looking at just the row *S*. Particularly, Sudoku puzzle can be trivially transform into an exact cover problem.

For the exact cover problem, one can consider a general situation that can be described abstractly as follows: Given a matrix of 0s and 1s, does it have a set of rows containing exactly one 1 in each column? For example, the matrix ***A***

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

has such a set (rows 1, 4, and 5). The columns are treated as elements of a universe, and the rows as subsets of the universe. Then the problem becomes to cover the universe with disjoint subsets. Or correspondingly the rows can be seen as elements of a universe, and the columns as subsets of that universe; then the problem is to find a collection of elements that intersect each subset at exactly one point.

As for the dance steps, one way to implement algorithm X is to represent each 1 in the matrix ***A*** as a data object *x* with five fields *L[x]*, *R[x], U[x], D[x], C[x].* Matrix's rows are doubly linked as circular lists via the *L(left)* and *R(right)* fields; columns are doubly linked as circular lists via the *U(up)* and *D(down)* fields. Each column list has a special data object called list header.

The column object contains list headers. Each column object *y* contains the fields *L[y], R[y], U[y], D[y],* and *C[y]* of a data object and two additional fields, *S[y]* (size) and *N[y]* (name). The size is the number of 1s, and the name is a symbolic identifier for printing the answers. The *C* field of each object points to the column object at the head of the relevant column.

The *L* and *R* fields of the list headers link together all columns that still need to be covered. This circular list also includes A special column object called root *h* is a part of this circular list, which is a master header for all the active headers. The fields *U[h], D[h], C[h], S[h],* and *N[h]* are not used.

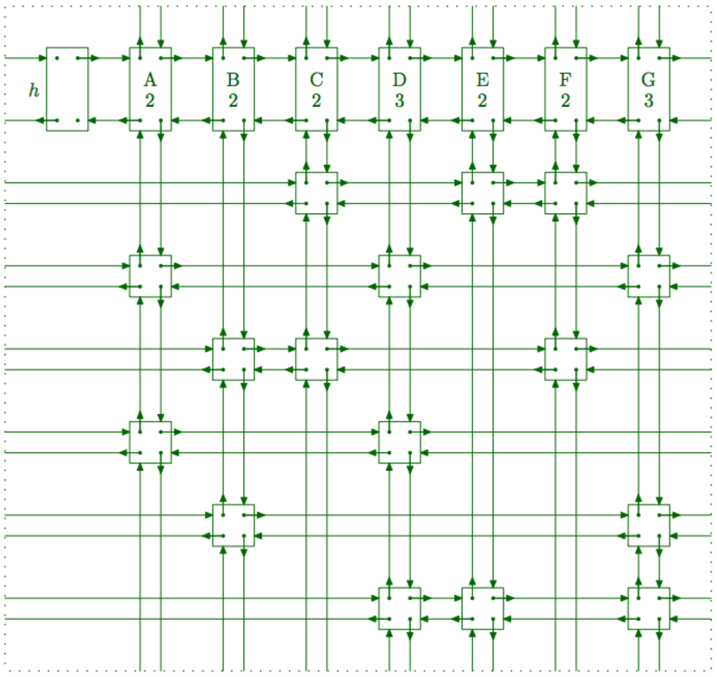


Figure 4. Four-way-linked representation of the exact cover problem

For example, the 0-1 matrix of ***A*** can be represented by the objects shown in figure 2, the columns are A, B, C, D, E, F, and G.

The nondeterministic algorithm can be executed as a recursive procedure *search(k)*, which is invoked initially with *k = 0*:

```
If $R[h] = h$, print the current solution and return.
Otherwise choose a column object c.
Cover column $c$.
For each $r \leftarrow D[c], D[D[c]], \ldots,$ while $r \neq c$,
    set $O_k \leftarrow r$;
    for each $j \leftarrow R[r], R[R[r]], \ldots,$ while $j \neq r$,
        cover column $j$;
search$(k + 1)$;
```

```
set $r \leftarrow O_k$ and $c \leftarrow C[r]$;
for each $j \leftarrow L[r], L\big[L[r]\big], \ldots$, while $j \neq r$,
    uncover column $j$.
Uncover column $c$ and return.
```

In the standard 16×16 Sudoku variant, there are four kinds of constraints:
Row-Column: Each intersection of a row and column, i.e, each cell, must contain exactly one number.
Row-Number: Each row must contain each number exactly once.
Column-Number: Each column must contain each number exactly once.
Box-Number: Each box must contain each number exactly once.

We need to convert these constraints into an exact cover problem matrix. The meaning of columns and rows represent different meanings. The column ($4\times16^2$) of the matrix has 4 constraints:
1. Position limit: Each position has exactly one number.
2. Column limit: Each column must contain each number exactly once.
3. Row limit: Each row must contain each number exactly once.
4. Area limit: Each area must contain each number exactly once.
The row ($16^3$) of the matrix represent to put each number to each box.
Then, use Dancing Link to solve the Sudoku problem.

## 3. Results

We used C++ to realize Algorithms Backtracking Search and Dancing Links. The following tablet shows the results of these two.
Machine: MacBook Pro (13-inch, Mid 2012)
Processor: 2.5 GHz Intel Core i5
Memory: 16G

**Backtracking Search with forward checking**

| Instance | Run time(s) | Number of search space nodes | Number of backtracks |
|---|---|---|---|
| Hexa-100 | 0.291882 | 196840 | 196635 |
| Hexa-101 | 1.53335 | 1021461 | 1021248 |
| Hexa-101(With MRV) | 0.2738 | 236 | 22 |
| Hexa-20(With MRV) | 8494.838 | 14651075 | 14650915 |
| Hexa-21, 50, 80, 81, 82, 100 | >10800(estimate) | Not Known | Not Known |

**Dancing Links**

| Instance | Hexa-20 | Hexa-21 | Hexa-50 | Hexa-80 | Hexa-81 | Hexa-82 | Hexa-100 | Hexa-101 |
|---|---|---|---|---|---|---|---|---|
| Run Time(ms) | 6.687 | 6.865 | 6.516 | 6.703 | 6.572 | 8.643 | 4.758 | 7.21 |

As shown above, due to varied search space of different Sudoku inputs, the run time varies a lot. Backtracking Search with forward checking and minimum remaining values performs better than Backtracking Search with forward checking. Dancing Links performs much better than Backtracking Search with forward checking. As for Hexa-21, 50, 80, 81, 82, 100, Backtracking Search with forward checking is too slow to return the results.

## 4. Conclusion

As we can see, Dancing Links performs much better than Backtracking Search with forward checking. We added minimum remaining values to Backtracking Search with forward checking to run faster. Also, there may be some ways to improve Dancing Links. Currently we search from the root. It may be more efficient to look for the unsolved column with the fewest elements and solve them in turn.

## Reference:

1. Solving Sudoku with Dancing Links. Retrieved from https://rafal.io/posts/solving-sudoku-with-dancing-links.html
2. Donald E. Knuth.(2000) Dancing Links. Retrieved from https://arxiv.org/pdf/cs/0011047v1.pdf
3. How to convert Sudoku problems to exact cover problems. Retrieved from https://www.zhihu.com/question/23022745
4. WanCangYiShu. Practice Algorithms: Using Dancing Links to Solve Sudoku. Retrieved from http://www.cnblogs.com/grenet/p/3163550.html
5. Luis E. Ortiz, Notes on Algorithms for Constraint Satisfaction Problems. Retrieved from:http://www.personal.umd.umich.edu/~leortiz/teaching/6.034f/Fall06/csp/csp_notes.pdf
6. Stuart Russell, Peter Norvig. Artificial Intelligence: A Modern Approach (Third Edition), 214-230

## Appendix

### Results from Dancing Links:

**Hexa-20:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 0 | 4 | B | F | 6 | 9 | C | D | E | A | 8 | 3 | 7 | 5 |
| 3 | E | C | B | 8 | D | A | 1 | F | 0 | 5 | 7 | 2 | 4 | 6 | 9 |
| A | 5 | 9 | 7 | 2 | 3 | E | 4 | 6 | 1 | 8 | B | D | 0 | F | C |
| 8 | F | D | 6 | C | 0 | 7 | 5 | 3 | 9 | 2 | 4 | 1 | B | A | E |
| 6 | B | 8 | 2 | 7 | 1 | F | E | 9 | 3 | A | D | 4 | 5 | C | 0 |
| 4 | A | 7 | 1 | 0 | 9 | 5 | 8 | 2 | E | 6 | C | 3 | D | B | F |
| E | 9 | 3 | 5 | 4 | 6 | C | D | B | F | 0 | 8 | A | 1 | 2 | 7 |
| 0 | D | F | C | 3 | B | 2 | A | 1 | 4 | 7 | 5 | 6 | 9 | E | 8 |
| B | C | 4 | E | A | 2 | D | F | 8 | 5 | 1 | 3 | 9 | 7 | 0 | 6 |
| 7 | 3 | 6 | F | 9 | E | 8 | 0 | D | A | B | 2 | 5 | C | 1 | 4 |
| 5 | 8 | A | 9 | 6 | 4 | 1 | 3 | 0 | 7 | C | F | B | E | D | 2 |
| 2 | 0 | 1 | D | 5 | C | B | 7 | E | 6 | 4 | 9 | F | A | 8 | 3 |
| C | 6 | 2 | 3 | E | A | 0 | B | 4 | 8 | 9 | 1 | 7 | F | 5 | D |
| D | 7 | B | 8 | F | 5 | 9 | 6 | A | C | 3 | E | 0 | 2 | 4 | 1 |
| 9 | 1 | E | 0 | D | 7 | 4 | 2 | 5 | B | F | 6 | C | 8 | 3 | A |
| F | 4 | 5 | A | 1 | 8 | 3 | C | 7 | 2 | D | 0 | E | 6 | 9 | B |

**The time spent for Dance Link: 0.006687 seconds.**

**Hexa-21:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 7 | 6 | 3 | 2 | 5 | 9 | F | B | C | 1 | 0 | 8 | 4 | D | E |
| C | E | 9 | F | B | 7 | 0 | 4 | 6 | D | 3 | 8 | A | 2 | 1 | 5 |
| 2 | 0 | D | B | 3 | 8 | 1 | 6 | A | 4 | 5 | E | F | 7 | 9 | C |
| 8 | 1 | 4 | 5 | D | E | A | C | 7 | F | 9 | 2 | 6 | 0 | 3 | B |
| D | F | C | 9 | A | 4 | E | 1 | 5 | 2 | 0 | 3 | B | 6 | 7 | 8 |
| 4 | A | 5 | 6 | F | 3 | 8 | 9 | C | 7 | B | D | 0 | E | 2 | 1 |
| 1 | 3 | 0 | 2 | 7 | 6 | B | 5 | F | 8 | E | A | 9 | D | C | 4 |
| 7 | B | 8 | E | 0 | 2 | C | D | 9 | 1 | 4 | 6 | 5 | A | F | 3 |
| 3 | 2 | A | 1 | 8 | 9 | 4 | 7 | E | B | D | F | C | 5 | 0 | 6 |
| 0 | 9 | B | 4 | E | C | 6 | 3 | 1 | 5 | 8 | 7 | D | F | A | 2 |
| F | 5 | E | C | 1 | 0 | D | B | 2 | A | 6 | 9 | 4 | 3 | 8 | 7 |
| 6 | 8 | 7 | D | 5 | F | 2 | A | 3 | 0 | C | 4 | 1 | B | E | 9 |
| B | 6 | 1 | 0 | 9 | D | F | E | 8 | 3 | 2 | 5 | 7 | C | 4 | A |
| E | C | 2 | A | 6 | 1 | 7 | 0 | 4 | 9 | F | B | 3 | 8 | 5 | D |
| 9 | D | 3 | 7 | 4 | B | 5 | 8 | 0 | E | A | C | 2 | 1 | 6 | F |
| 5 | 4 | F | 8 | C | A | 3 | 2 | D | 6 | 7 | 1 | E | 9 | B | 0 |

**The time spent for Dance Link: 0.006865 seconds.**

**Hexa-50:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 2 | A | 7 | 6 | D | 3 | F | 0 | B | 5 | E | 8 | 9 | C |
| B | 8 | 3 | D | 5 | F | C | E | 7 | 1 | 6 | 9 | A | 0 | 2 | 4 |
| 7 | F | 5 | E | 0 | 4 | 9 | A | C | 3 | 2 | 8 | D | 6 | 1 | B |
| 0 | 9 | C | 6 | B | 2 | 8 | 1 | D | 4 | E | A | 5 | F | 3 | 7 |
| A | 6 | 1 | F | C | E | 3 | 5 | 4 | 9 | D | 7 | 0 | B | 8 | 2 |
| 3 | 7 | 8 | C | 2 | D | 6 | 9 | 0 | A | 5 | B | 4 | 1 | E | F |
| D | 4 | 0 | 5 | 8 | 1 | F | B | 2 | 6 | C | E | 9 | 7 | A | 3 |
| E | B | 9 | 2 | 4 | A | 0 | 7 | 3 | 8 | 1 | F | 6 | C | D | 5 |
| 1 | 3 | F | 7 | E | 9 | 5 | 8 | B | D | 0 | 4 | 2 | A | C | 6 |
| 2 | C | B | 9 | A | 7 | 4 | 0 | 5 | E | 8 | 6 | 3 | D | F | 1 |
| 6 | D | E | 4 | 1 | 3 | B | F | A | 2 | 7 | C | 8 | 5 | 0 | 9 |
| 8 | 5 | A | 0 | D | C | 2 | 6 | 9 | F | 3 | 1 | B | 4 | 7 | E |
| C | E | D | B | 9 | 0 | A | 4 | 6 | 7 | F | 2 | 1 | 3 | 5 | 8 |
| 5 | A | 4 | 1 | 3 | B | 7 | 2 | 8 | C | 9 | 0 | F | E | 6 | D |
| 9 | 0 | 6 | 3 | F | 8 | E | C | 1 | 5 | 4 | D | 7 | 2 | B | A |
| F | 2 | 7 | 8 | 6 | 5 | 1 | D | E | B | A | 3 | C | 9 | 4 | 0 |

**The time spent for Dance Link: 0.006516 seconds.**

**Hexa-80:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | E | F | 0 | 5 | 3 | 9 | 2 | 1 | 8 | C | B | D | 6 | 7 | 4 |
| 3 | 9 | D | 4 | C | B | E | 6 | F | 2 | 5 | 7 | A | 1 | 0 | 8 |
| 6 | C | 5 | 8 | 1 | 4 | 0 | 7 | A | D | 9 | E | 2 | B | 3 | F |
| 1 | 7 | 2 | B | A | D | F | 8 | 6 | 3 | 4 | 0 | 9 | E | 5 | C |
| 9 | 5 | 8 | 6 | E | 2 | 7 | F | C | B | 1 | A | 3 | D | 4 | 0 |
| B | F | C | E | 9 | 6 | 5 | A | D | 0 | 3 | 4 | 8 | 7 | 2 | 1 |
| 4 | 2 | 1 | A | 3 | 0 | C | D | 5 | 7 | 6 | 8 | F | 9 | E | B |
| 7 | 0 | 3 | D | 8 | 1 | B | 4 | 2 | F | E | 9 | C | 5 | A | 6 |
| 2 | B | 4 | C | 6 | E | 3 | 5 | 9 | 1 | 7 | D | 0 | F | 8 | A |
| E | A | 9 | 3 | 2 | 8 | D | C | B | 6 | 0 | F | 5 | 4 | 1 | 7 |
| F | 1 | 0 | 5 | B | 7 | A | 9 | 4 | E | 8 | 2 | 6 | C | D | 3 |
| 8 | D | 6 | 7 | 4 | F | 1 | 0 | 3 | 5 | A | C | B | 2 | 9 | E |
| C | 3 | E | 1 | D | A | 6 | B | 8 | 4 | 2 | 5 | 7 | 0 | F | 9 |
| 5 | 6 | 7 | 9 | F | C | 4 | E | 0 | A | D | 3 | 1 | 8 | B | 2 |
| 0 | 8 | A | F | 7 | 5 | 2 | 1 | E | 9 | B | 6 | 4 | 3 | C | D |
| D | 4 | B | 2 | 0 | 9 | 8 | 3 | 7 | C | F | 1 | E | A | 6 | 5 |

**The time spent for Dance Link: 0.006703 seconds.**

**Hexa-81:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 9 | 3 | 7 | 4 | D | 1 | 2 | 8 | F | 6 | B | E | 5 | 0 | C |
| 6 | 0 | E | F | 8 | 9 | B | A | 5 | 2 | 7 | C | D | 1 | 4 | 3 |
| D | 2 | 1 | B | 3 | 6 | C | 5 | E | 0 | 4 | A | 7 | F | 8 | 9 |
| C | 4 | 8 | 5 | E | 0 | F | 7 | 1 | 3 | 9 | D | 6 | B | A | 2 |
| 8 | B | 9 | E | 0 | 3 | 7 | 4 | C | 5 | D | F | 2 | A | 6 | 1 |
| F | 1 | D | C | 5 | A | 2 | 6 | 0 | 7 | B | 9 | 3 | 8 | E | 4 |
| 2 | 5 | 4 | 0 | F | 1 | 8 | D | 3 | 6 | A | E | 9 | 7 | C | B |
| 7 | 3 | A | 6 | C | E | 9 | B | 2 | 4 | 1 | 8 | 5 | D | F | 0 |
| 4 | 7 | 5 | 2 | A | F | 0 | E | D | 8 | 3 | 1 | B | C | 9 | 6 |
| 0 | E | B | A | 6 | 8 | D | C | 7 | 9 | F | 4 | 1 | 2 | 3 | 5 |
| 3 | D | F | 1 | 2 | B | 5 | 9 | 6 | A | C | 0 | 4 | E | 7 | 8 |
| 9 | 6 | C | 8 | 7 | 4 | 3 | 1 | B | E | 5 | 2 | A | 0 | D | F |
| E | 8 | 7 | 3 | D | C | 6 | F | 4 | B | 2 | 5 | 0 | 9 | 1 | A |
| 1 | A | 6 | 4 | B | 5 | E | 0 | 9 | C | 8 | 7 | F | 3 | 2 | D |
| B | C | 2 | 9 | 1 | 7 | A | 3 | F | D | 0 | 6 | 8 | 4 | 5 | E |
| 5 | F | 0 | D | 9 | 2 | 4 | 8 | A | 1 | E | 3 | C | 6 | B | 7 |

**The time spent for Dance Link: 0.006572 seconds.**

**Hexa-82:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | D | 6 | 2 | A | 5 | C | 7 | E | 4 | 0 | 3 | F | 1 | B | 9 |
| 1 | 7 | 4 | 3 | 8 | 2 | D | 9 | B | 6 | A | F | 0 | 5 | E | C |
| 0 | E | 5 | F | 1 | B | 4 | 3 | 9 | D | 8 | C | 6 | A | 7 | 2 |
| B | C | A | 9 | 6 | E | 0 | F | 2 | 1 | 7 | 5 | 8 | D | 4 | 3 |
| 5 | F | 9 | B | 7 | 6 | A | 2 | 3 | C | D | 8 | E | 4 | 0 | 1 |
| D | 6 | E | 4 | C | F | 3 | 5 | 1 | 0 | B | 2 | 9 | 8 | A | 7 |
| 7 | 2 | 0 | A | D | 1 | 8 | E | F | 9 | 4 | 6 | B | C | 3 | 5 |
| C | 8 | 3 | 1 | B | 4 | 9 | 0 | 7 | A | 5 | E | 2 | F | 6 | D |
| E | A | 8 | 6 | 9 | 3 | B | D | C | 2 | F | 0 | 5 | 7 | 1 | 4 |
| F | 0 | 2 | C | E | 7 | 5 | 4 | 8 | 3 | 6 | 1 | A | 9 | D | B |
| 4 | 3 | 7 | 5 | 0 | 8 | 1 | 6 | A | B | 9 | D | C | E | 2 | F |
| 9 | 1 | B | D | F | C | 2 | A | 4 | 5 | E | 7 | 3 | 6 | 8 | 0 |
| 2 | 9 | C | 8 | 5 | D | 7 | B | 0 | E | 1 | A | 4 | 3 | F | 6 |
| 3 | 5 | F | E | 4 | 0 | 6 | 8 | D | 7 | 2 | 9 | 1 | B | C | A |
| A | 4 | D | 0 | 3 | 9 | E | 1 | 6 | F | C | B | 7 | 2 | 5 | 8 |
| 6 | B | 1 | 7 | 2 | A | F | C | 5 | 8 | 3 | 4 | D | 0 | 9 | E |

**The time spent for Dance Link: 0.008643 seconds.**

**Hexa-100:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 0 | E | 3 | 4 | 8 | A | D | 5 | 9 | 1 | B | F | 6 | C | 2 |
| 6 | B | 1 | 8 | E | C | 5 | F | 7 | 0 | 3 | 2 | 9 | A | 4 | D |
| 5 | 4 | A | D | 2 | 9 | B | 3 | F | 6 | C | E | 0 | 8 | 7 | 1 |
| 2 | C | F | 9 | 6 | 7 | 0 | 1 | A | 8 | 4 | D | E | 3 | 5 | B |
| D | E | 8 | 7 | B | 3 | F | 0 | 1 | 2 | 9 | A | 5 | 4 | 6 | C |
| 0 | 6 | C | 2 | A | E | 1 | 4 | D | 5 | 8 | 7 | B | F | 9 | 3 |
| 4 | 5 | 9 | B | C | D | 2 | 7 | 0 | 3 | 6 | F | 8 | 1 | A | E |
| 1 | F | 3 | A | 9 | 6 | 8 | 5 | C | E | B | 4 | D | 0 | 2 | 7 |
| A | D | 4 | 6 | 0 | 2 | E | B | 3 | 1 | 7 | 9 | C | 5 | F | 8 |
| 8 | 3 | 0 | 5 | D | F | 7 | 6 | 2 | B | A | C | 4 | E | 1 | 9 |
| 9 | 1 | B | F | 3 | 5 | 4 | C | 6 | D | E | 8 | 7 | 2 | 0 | A |
| E | 7 | 2 | C | 1 | A | 9 | 8 | 4 | F | 0 | 5 | 3 | B | D | 6 |
| F | 2 | D | E | 7 | B | 6 | A | 9 | 4 | 5 | 3 | 1 | C | 8 | 0 |
| B | A | 6 | 0 | 5 | 4 | C | E | 8 | 7 | D | 1 | 2 | 9 | 3 | F |
| 3 | 9 | 5 | 1 | 8 | 0 | D | 2 | B | C | F | 6 | A | 7 | E | 4 |
| C | 8 | 7 | 4 | F | 1 | 3 | 9 | E | A | 2 | 0 | 6 | D | B | 5 |

**The time spent for Dance Link: 0.004758 seconds.**

**Hexa-101:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7 | B | E | A | 6 | D | 4 | F | 8 | 9 | 1 | C | 3 | 2 | 5 |
| C | 9 | 3 | F | 2 | 8 | 5 | E | 7 | 6 | A | 0 | 4 | 1 | D | B |
| 1 | 5 | 6 | A | 7 | C | 0 | 3 | D | 2 | B | 4 | F | 9 | E | 8 |
| 2 | 4 | 8 | D | 1 | B | F | 9 | C | E | 5 | 3 | 6 | 0 | 7 | A |
| E | 8 | 1 | 3 | 9 | 7 | C | D | A | B | 6 | 5 | 0 | 2 | 4 | F |
| B | C | D | 4 | E | A | 6 | 5 | 8 | F | 0 | 2 | 9 | 7 | 1 | 3 |
| 9 | A | 5 | 2 | 3 | 0 | B | F | 4 | 1 | D | 7 | E | 8 | 6 | C |
| 6 | 0 | F | 7 | 4 | 2 | 8 | 1 | 3 | 9 | C | E | A | 5 | B | D |
| F | D | 2 | 9 | 0 | 3 | E | 6 | 5 | A | 1 | 8 | 7 | B | C | 4 |
| 5 | 6 | 7 | C | 8 | 9 | 4 | 2 | 0 | D | E | B | 3 | A | F | 1 |
| A | E | 4 | B | 5 | F | 1 | C | 2 | 3 | 7 | 6 | 8 | D | 0 | 9 |
| 3 | 1 | 0 | 8 | B | D | A | 7 | 9 | C | 4 | F | 2 | E | 5 | 6 |
| 7 | F | A | 5 | 6 | E | 9 | B | 1 | 0 | 8 | C | D | 4 | 3 | 2 |
| D | 3 | E | 6 | F | 5 | 7 | A | B | 4 | 2 | 9 | 1 | C | 8 | 0 |
| 8 | 2 | C | 1 | D | 4 | 3 | 0 | E | 5 | F | A | B | 6 | 9 | 7 |
| 4 | B | 9 | 0 | C | 1 | 2 | 8 | 6 | 7 | 3 | D | 5 | F | A | E |

**The time spent for Dance Link: 0.00721 seconds.**