SAS and Sudoku

Presented by: Steve First
2997 Yarmouth Greenway Drive, Madison, WI  53711
Phone: (608) 278-9964 • Web: www.sys-seminar.com
"The picture's pretty bleak, gentlemen. … The world's climates are changing, the mammals are taking over, and we all have a brain about the size of a walnut."
What is Sudoku?

• Latin Square problem, plus quadrants
• Subset of set theory, “exact cover problem”
• Simple to get started
• Many levels of difficulty
• Wildly popular
• Very addictive.
Sudoku Rules

- Each row, column, quadrant must use digits 1-9 one time.
Sudoku Strategies

- Algorithms
- Guess or trial and error
Why SAS?

• I know SAS
• Interesting array capabilities
• Limited “list processing” facilities.
Why write a Sudoku program?

• Interesting challenge
• Donald Knuths ‘Algorithm X’ using “dancing Links”
My Program

- Uses arrays to solve
- Uses “naked single” and “hidden single” algorithms
- Can add new algorithms in future
- Will solve simple puzzles
- Will not solve many difficult puzzles.
Unknown to most historians, William Tell had an older and less fortunate son named Warren.
My sudoku Program

/* sudoku program that implements naked and hidden single algorithm */
%let graphics=y;                      /* graphics available? */
/* include annotate macros */
%inc 'C:\Program Files\SAS\SAS 9.1\core\sasmacro\annomac.sas';

goptions hpos=60 vpos=36 target=winprtc; /* graphics options */
options mprint;
options nodate nonumber ls=132;title;  /* system options */
data sudoku;                          /* create two datasets */
   annods(keep=pagenochar STYLE FUNCTION COLOR XSYS YSYS HSYS
         WHEN POSITION X Y size linetype LINE text)
   ;
*infile 'C:\Documents and Settings\sfirst\Desktop\sudoku\oct15_lvl2.dat';
infile 'C:\Documents and Settings\sfirst\Desktop\sudoku\sep7_lvl4.dat';

/* define 81 variables that initial contain 9 digits each */
length r1c1 r1c2 r1c3 r1c4 r1c5 r1c6 r1c7 r1c8 r1c9 $9;
length r2c1 r2c2 r2c3 r2c4 r2c5 r2c6 r2c7 r2c8 r2c9 $9;
length r3c1 r3c2 r3c3 r3c4 r3c5 r3c6 r3c7 r3c8 r3c9 $9;
length r4c1 r4c2 r4c3 r4c4 r4c5 r4c6 r4c7 r4c8 r4c9 $9;
length r5c1 r5c2 r5c3 r5c4 r5c5 r5c6 r5c7 r5c8 r5c9 $9;
length r6c1 r6c2 r6c3 r6c4 r6c5 r6c6 r6c7 r6c8 r6c9 $9;
length r7c1 r7c2 r7c3 r7c4 r7c5 r7c6 r7c7 r7c8 r7c9 $9;
length r8c1 r8c2 r8c3 r8c4 r8c5 r8c6 r8c7 r8c8 r8c9 $9;
length r9c1 r9c2 r9c3 r9c4 r9c5 r9c6 r9c7 r9c8 r9c9 $9;

Array all81{9,9} $ r1c1--r9c9;       /* redefine all 81 vars into an array */
My sudoku Program

Array rowa {9,9} $ r1c1--r1c9  /* redefine all 81 vars into row array */
    r2c1--r2c9
    r3c1--r3c9
    r4c1--r4c9
    r5c1--r5c9
    r6c1--r6c9
    r7c1--r7c9
    r8c1--r8c9
    r9c1--r9c9;

    /* redefine all 81 vars into column array */
Array cola{9,9} $ r1c1  r2c1  r3c1  r4c1  r5c1  r6c1  r7c1  r8c1  r9c1
    r1c2  r2c2  r3c2  r4c2  r5c2  r6c2  r7c2  r8c2  r9c2
    r1c3  r2c3  r3c3  r4c3  r5c3  r6c3  r7c3  r8c3  r9c3
    r1c4  r2c4  r3c4  r4c4  r5c4  r6c4  r7c4  r8c4  r9c4
    r1c5  r2c5  r3c5  r4c5  r5c5  r6c5  r7c5  r8c5  r9c5
    r1c6  r2c6  r3c6  r4c6  r5c6  r6c6  r7c6  r8c6  r9c6
    r1c7  r2c7  r3c7  r4c7  r5c7  r6c7  r7c7  r8c7  r9c7
    r1c8  r2c8  r3c8  r4c8  r5c8  r6c8  r7c8  r8c8  r9c8
    r1c9  r2c9  r3c9  r4c9  r5c9  r6c9  r7c9  r8c9  r9c9;

    /* redefine all 81 vars into quad array */
Array quada{9,9}$ r1c1  r1c2  r1c3  r2c1  r2c2  r2c3  r3c1  r3c2  r3c3
    r1c4  r1c5  r1c6  r2c4  r2c5  r2c6  r3c4  r3c5  r3c6
    r1c7  r1c8  r1c9  r2c7  r2c8  r2c9  r3c7  r3c8  r3c9
    r4c1  r4c2  r4c3  r5c1  r5c2  r5c3  r6c1  r6c2  r6c3
    r4c4  r4c5  r4c6  r5c4  r5c5  r5c6  r6c4  r6c5  r6c6
    r4c7  r4c8  r4c9  r5c7  r5c8  r5c9  r6c7  r6c8  r6c9
    r7c1  r7c2  r7c3  r8c1  r8c2  r8c3  r9c1  r9c2  r9c3
    r7c4  r7c5  r7c6  r8c4  r8c5  r8c6  r9c4  r9c5  r9c6
    r7c7  r7c8  r7c9  r8c7  r8c8  r8c9  r9c7  r9c8  r9c9;
My sudoku Program

length work81 $ 81 ; /* 81 byte work area to contain one structure */
array cnta{9} cnt_1-cnt_9; /* array to count digit occurences */

link load_initial; /* mainline portion */

change='Y'; /* load initial puzzle */
do until(change='N' ); /* keep going until no more changes*/
  change='N'; /* change flag to no */
  i+1; /* times looped thru */
do Structure='R','C','Q'; /* do rows, cols, then quads */
do struct_no = 1 to 9; /* do all 9 rows, cols, quads */
  Link loadwork81; /* move structure to work area */
  link naked_singles; /* look for naked singles */
  link hidden_singles; /* then look for hidden singles */
  link unloadwork81; /* move work area back to arrays */
end; /* end of inner loop */
/* title with interation, struct */
rttitle= 'Interation ' !! put(i,3.) !! ' ' !!
           structure !! ' Structure Report';
link report; /* print results so far */
end; /* next structure */
end; /* next loop if still changed */
return; /* end of mainline */
My sudoku Program

load_initial:                        /* read in puzzle, init blnk cells */
do row=1 to 9;                       /* read 9 rows */
do col=1 to 9;                      /* and 9 columns */
    input all81{row,col} $char1. @;   /* input array element from file */
    if all81{row,col} in('.',' ') then/* if missing or blank? */
        all81{row,col}='123456789';    /* initialize to all 9 possibles */
    end;                                /* end of col loop */
    input;                              /* clear input pointer */
end;                                 /* next row */
rtitle='Step1 - initial load '; /* set title */
link report;                         /* print initial puzzle */
return;                              /* end of load */

Loadwork81:                          /* load work with cells from r,c,q */
do cell=1 to 9;                     /* 9 cells, each 9 digits long */
    fposition=((cell-1)*9)+1;        /* cell 1 to pos 1, 2 to 10 etc */
    if structure="Q" then            /* if quad? */
        cell_in=quada(struct_no,cell); /* move cell from quad array */
    else
        if structure="R" then        /* same for rows */
            cell_in=rowa(struct_no,cell);
        else
            if structure="C" then    /* and cols */
                cell_in=cola(struct_no,cell);
            end;
        end;                        /* next cell */
save_work81=work81;                 /* save work to look for changes */
return;                              /* end of load work */
My sudoku Program

/***************************************************************************/
/* if cell contains a single candidate, then eliminate candidate */
/* from all cells in structure. */
/***************************************************************************/
naked_singles:
  do cell=1 to 9;
    fposition=((cell-1)*9)+1;
    search=left(substr(work81,fposition,9));
    if length(search)=1 then                /* single dig? */
      do;
        work81=translate(work81,' ',search); /* change all cells inc itself */
        substr(work81,fposition,9)=search; /* set single digt back to search*/
      end;
  end;
return;

/***************************************************************************/
/* a digit occurs only once in a structure, but may have other digs */
/* with it. when found, set cell and eliminate in rest of structure*/
/***************************************************************************/
hidden_singles:
  do digit=1 to 9;
    digit_char=put(digit,1.);
    cnta{digit}=count(work81,digit_char);
    if cnta{digit}= 1 then
      do;
        fposition=index(work81,digit_char);
        fposition=(int((fposition-1)/9)*9)+1;
        substr(work81,fposition,9)=digit_char !!'        ';
      end;
  end;
return;
unloadwork81:                /* reload structure with changed values */
   do cell=1 to 9;
      fposition=((cell-1)*9)+1;
      if structure="Q" then
         quada{struct_no,cell}=substr(work81,fposition,9);
      else
         if structure="R" then
            rowa{struct_no,cell}=substr(work81,fposition,9);
         else
            if structure="C" then
               cola{struct_no,cell}=substr(work81,fposition,9);
      end;
      if work81 ne save_work81     /* was any change made? */
         then change='Y';          /* yes, set flag */
      return;

report:                        /* graphic report */
   if "+graphics" ne "y" then
      goto report_text;
   pageno+1;
   pagenochar='Page_ ' !! put(PAGENO,Z2.);
   %DCLANNO ;

      x=3;       /* title */
      y=32;
      text=rtitle;
      style='Swissb';
      size=1.5;
      position='6';
      output;
My sudoku Program

DO Y=1 TO 28 by 3;
    if mod(y-1,9)= 0 then size=10;
    else
        if mod(y-1,3)= 0 then size=3;
        else size=1;
    if size=3 then linetype=2;
    else
        linetype=1;
    %LINE(1,Y,28,Y,BLACK,linetype,size);
END;
DO X=1 TO 28 by 3;
    if mod(x-1,9)= 0 then size=10;
    else
        if mod(x-1,3)= 0 then size=3;
        else size=1;
    if size=3 then
        linetype=2;
    else
        linetype=1;
    %LINE(X,1,X,28,BLACK,linetype,size);
END;

Do row=1 to 9;
do col=1 to 9;
    x=3*col - .5; /* trans to lower left origin on 27*27 grid */
    y=-3*row + 29.5;
    savex=x;
    savey=y;
    function='label';
    Color='Black';


My sudoku Program

```plaintext
style='swissb';
  position='+';
  search=left(all81{row,col});
if length(search)=1 then    /* single dig? */
  do;
    text=substr(search,1,1);
    size=2;
    output;
  end;
else
  do fposition=1 to length(search);
    text=substr(search,fposition,1);
    size=.8;
    style='swiss';
    color='Red';
    x=savex;
    y=savey;
    if text in ('1','2','3') then
      y=y+1;
    if text in ('7','8','9') then
      y=y-1;
    if text in ('1','4','7') then
      x=x-1;
    if text in ('3','6','9') then
      x=x+1;
    output;
  end;
end;
end;
```
My sudoku Program

report_text: /* text report */
file print;
put _page_@;
put rtitle;
put @1 110'*'*';
do row=1 to 9;
   do col=1 to 9;
      if col in(1,4,7) then
         put '* '@;
      else
         put '|' '@;
      put +1 all81{row,col} $char9. +1 '@;
   end;
   put @110 '*';
   if row in (3,6,9) then
      put @1 110'*'*';
   else
      do;
         put 3'*' *-------- | *-------- | *-------- ' '@;
         put @110 '*';
      end;
   end;
end;
put ///;
return;
run;

PROC GREPLAY NOFS IGOUT=WORK.GSEG;
   DELETE _ALL_; QUIT;
PROC Ganno ANNOTATE=annods
   name=pagenoCHAR;TITLE;run;
A Puzzle That Program Can Solve

A level 2 Sudoku.

..5.9.8.6
42....... 
.3.6.....5 
.7.9....... 
..97521... .....8.7. 
......8.7. 6....4.5. 
........29  
8.7.2.6..
Program Output

Step 1 — initial load

```
1 2 3 1 2 3 1 2 3
4 5 6 4 5 6 4 5 6
7 8 9 7 8 9 7 8 9

4 2 3
5 6 9
8 7 1

4 2 3
5 6 9
8 7 1

4 2 3
5 6 9
8 7 1
```

Systems Seminar Consultants, Inc
Program Output

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1 R Structure Report</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Program Output

Interation 1 C Structure Report

![Program Output Diagram]
Program Output

Interaction 1 Q Structure Report

```
    1   15   29   8   6
   4  2  6  9  3  8  6  4
  9  3  8  6  4  1  2  5
```

Systems Seminar Consultants, Inc
Program Output

Interation  2  R  Structure Report

```
7 1 5
4 2 6
9 3 8
```

```
2 9 3
1 5 3
6 4 7
```

```
8 4 6
3 1 3
2 1 5
```

```
1 2 3
4 6 8
9 4 6
```

```
3 4 6
1 4 6
5 4 6
```

```
9 7 5 2
1 4 6
8 9 7
```

```
6 9 2
1 3 1
5 1 3
```

```
4 3 5
3 4 5
3 4 5
2 9
```

```
8 7 3
1 4 5
```

```
```
Program Output

**Iteration 2 C Structure Report**

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>1</th>
<th>5</th>
<th></th>
<th>2</th>
<th>9</th>
<th>3</th>
<th></th>
<th>8</th>
<th>4</th>
<th>6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
<td>3</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
<td>6</td>
<td>4</td>
<td>7</td>
<td></td>
<td>2</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td>7</td>
<td></td>
<td>1</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
<td>7</td>
<td>5</td>
<td>2</td>
<td></td>
<td>1</td>
<td></td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>3</td>
<td></td>
<td>8</td>
<td></td>
<td>9</td>
<td></td>
<td>7</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td>1</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td>7</td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Systems Seminar Consultants, Inc
Program Output

Interation  2  Q  Structure  Report

1  1  5  9  3  8  4  6
2  4  6  1  5  1  8  5
3  2  9  6  4  7  3  9  7
4  9  8  3  1  8  1  5
5  7  4  6  4  8  2  3
6  1  5  2  3  8  6  4
7  3  8  9  7  5  2  1  4
8  6  4  8  9  7  2  3
9  1  3  2  3  6  5  6
10  3  1  3  1  6  5  6
11  4  2  9  6  3  1
Program Output

Interation 3 R Structure Report

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

27
Program Output (Note Naked Pair)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>18</td>
<td>5</td>
<td>39</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>47</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>58</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>52</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>89</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>75</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>42</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>29</td>
<td>63</td>
</tr>
</tbody>
</table>

Interation 3 C Structure Report
Program Output

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Q Structure Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 1 5</td>
<td></td>
</tr>
<tr>
<td>4 2 6</td>
<td></td>
</tr>
<tr>
<td>9 3 8</td>
<td></td>
</tr>
<tr>
<td>2 7 4</td>
<td></td>
</tr>
<tr>
<td>3 8 9</td>
<td></td>
</tr>
<tr>
<td>5 6 1</td>
<td></td>
</tr>
<tr>
<td>6 9 2</td>
<td></td>
</tr>
<tr>
<td>1 5 3</td>
<td></td>
</tr>
<tr>
<td>8 4 7</td>
<td></td>
</tr>
</tbody>
</table>

2 3
Puzzle Solved

**Interation 4 R Structure Report**

<table>
<thead>
<tr>
<th>7</th>
<th>1</th>
<th>5</th>
<th>2</th>
<th>9</th>
<th>3</th>
<th>8</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
One, two... testing... one, two...

Hey!... Hey, you! You little bug-eyed greasy sardine! Let me tell you something about your sister!...

Hello, hello... Al, is this thing on?

Testing whether fish have feelings
Other Algorithms

- Naked pairs, triples, quads
- Hidden pairs, triples, quads
- X-wing
- Swordfish, jellyfish
- Color trap, wrap, wing
- Empty rectangle, hinge
- Remote pair
Algorithm X and Dancing Links

A trial and error solver by Donald Knuth.

If $x$ points to doubly linked list, $L[x]$, $R[x]$ point to predecessor, successor.

Then $L[R[x]] \leftarrow L[x]$, $R[L[x]] \leftarrow R[x]$ removes $x$.

And

$L[R[x]] \leftarrow x$, $R[L[x]] \leftarrow x$

Puts $x$ back In the list.
Resources

- Wikipedia
- Suducue.net
- Many, others
"Oh, lovely — just the hundredth time you’ve managed to cut everyone’s head off."
Contact Us

Systems Seminar Consultants, Inc.
SAS® Training, Consulting, & Help Desk Services
2997 Yarmouth Greenway Drive • Madison, WI  53711
(608) 278-9964 • Fax (608) 278-0065
www.sys-seminar.com

Steve First, President
608-278-9964 x 306
sfirst@sys-seminar.com