Operating System Project #2

2017012351 이영섭

내용

Sudoku 알고리즘

1. void *check_rows(void *arg)
   A. 각 행의 원소를 확인하면서 compare_rows의 원소와 비교하여 같으면 0으로 치환해주고 rows_cnt++를 해준다.
   B. 행에 1~9의 모든 원소가 존재한다면 rows_cnt 값이 9가 될 것이고, 9가 되면 valid[0][i] 값을 1로 바꿔준다.
   C. 행에 1~9의 모든 원소가 존재하지 않는다면 rows_cnt 값이 9가 아닐 것이고, 9가 아니면 valid[0][i] 값을 0으로 바꿔준다.
2. void *check_columns(void *arg)
   A. 각 열의 원소를 확인하면서 compare_columns의 원소와 비교하여 같으면 0으로 치환해주고 columns_cnt++를 해준다.
   B. 행에 1~9의 모든 원소가 존재한다면 columns_cnt 값이 9가 될 것이고, 9가 되면 valid[1][i] 값을 1로 바꿔준다.
   C. 행에 1~9의 모든 원소가 존재하지 않는다면 columns_cnt 값이 9가 아닐 것이고, 9가 아니면 valid[1][i] 값을 0으로 바꿔준다.
3. void *check_subgrid(void *arg)
   A. location_t 구조체를 입력으로 받아 row와 column을 변수로 선언
   B. 각 subgrid의 원소를 확인하면서 compare_subgrids의 원소와 비교하여 같으면 0으로 치환해주고 subgrids_cnt++를 해준다.
   C. subgrid에 1~9의 모든 원소가 존재한다면 subgrids_cnt 값이 9가 될 것이고, 9가 되면 valid[2][subgrids_num] 값을 1로 바꿔준다.
   D. subgrid에 1~9의 모든 원소가 존재하지 않는다면 subgrids_cnt 값이 9가 아닐 것이고, 9가 아니면 valid[2][subgrids_num] 값을 0으로 바꿔준다.
4. void check_sudoku(void)
   A. check_rows() thread 생성
   B. check_columns() thread 생성
   C. 구조체 선언 해주고 9개의 check_subgrids() thread 생성
   D. 기다렸다가 pthread_join()로 thread 종료

프로그램 소스파일

```c
/*
 * Copyright 2021. Heekuck Oh, all rights reserved
 * 이 프로그램은 한양대학교 ERICA 소프트웨어학부 재학생을 위한 교육용으로 제작되었습니다.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>

typedef struct{                         //구조체 선언
  int row;
  int column;
}location_t;

/*
 * 기본 스도쿠 퍼즐
 */
int sudoku[9][9] =
{{6,3,9,8,4,1,2,7,5},{7,2,4,9,5,3,1,6,8},{1,8,5,7,2,6,3,9,4},{2,5,6,1,3,7,4,8,9},{4
,9,1,5,8,2,6,3,7},{8,7,3,4,6,9,5,2,1},{5,4,2,3,9,8,7,1,6},{3,1,8,6,7,5,9,4,2},{9,6,
7,2,1,4,8,5,3}};

/*
 * valid[0][0], valid[0][1], ..., valid[0][8]: 각 행이 올바르면 1, 아니면 0
 * valid[1][0], valid[1][1], ..., valid[1][8]: 각 열이 올바르면 1, 아니면 0
 * valid[2][0], valid[2][1], ..., valid[2][8]: 각 3x3 그리드가 올바르면 1, 아니면 0
 */
int valid[3][9];

/*
 * 스도쿠 퍼즐의 각 행이 올바른지 검사한다.
 * 행 번호는 0부터 시작하며, i번 행이 올바르면 valid[0][i]에 1을 기록한다.
 */
void *check_rows(void *arg)
{
  for(int i = 0; i < 9; i++){
    int rows_cnt = 0;
    int compare_rows[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9};  //row 확인을 위한 배열
    for(int j = 0; j < 9; j++){
      for(int k = 0; k < 9; k++){
        if(sudoku[i][j] == compare_rows[k]){        //현재 원소와 비교용 배열과 비교
          compare_rows[k] = 0;                      //같으면 비교용 배열을 0으로 치환
          rows_cnt++;                               //하고 rows_cnt 값을 1 증가
        }
      }
    }
    if(rows_cnt == 9) valid[0][i] = 1;   //rows_cnt 값이 9이면 valid[0][i]값을 1로 바꿈
    else valid[0][i] = 0;                //9가 아니라면 0으로 바꿈
  }
  pthread_exit(NULL);                     //프로세스 종료
```

```c
}

/*
 * 스도쿠 퍼즐의 각 열이 올바른지 검사한다.
 * 열 번호는 0 부터 시작하며, j 번 열이 올바르면 valid[1][j]에 1 을 기록한다.
 */
void *check_columns(void *arg)
{
  for(int j = 0; j < 9; i++){
    int compare_columns[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9}; //column 확인을 위한 배열
    int columns_cnt = 0;
    for(int i = 0; i < 9; j++){
      for(int k = 0; k < 9; k++){
        if(sudoku[i][j] == compare_columns[k]){     //현재 원소와 비교용 배열과 비교
          compare_columns[k] = 0;                   //같으면 비교용 배열을 0 으로 치환
          columns_cnt++;                            //하고 columns_cnt 값을 1 증가
        }
      }
    }
    if(columns_cnt == 9) valid[1][j] = 1;
       //columns_cnt 값이 9 이면 valid[1][j]값을 1 로 바꿈
    else valid[1][j] = 0;    //9 가 아니라면 valid[1][j]값을 0 으로 바꿈
  }
  pthread_exit(NULL);       //프로세스 종료
}

/*
 * 스도쿠 퍼즐의 각 3x3 서브그리드가 올바른지 검사한다.
 * 3x3 서브그리드 번호는 0 부터 시작하며, 왼쪽에서 오른쪽으로, 위에서 아래로 증가한다.
 * k 번 서브그리드가 올바르면 valid[2][k]에 1 을 기록한다.
 */
void *check_subgrid(void *arg)
{
  location_t *t1 = (location_t *)arg;    //location_t 를 t1 으로 받아
  int row = t1 -> row;                   //row 를 t1 의 row 로 선언
  int column = t1 -> column;             //column 을 t1 의 column 으로 선언
  int compare_subgrids[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9};   //subgrid 확인을 위한 배열
  int subgrids_num, subgrids_cnt = 0;
  for(int i = row*3; i < row*3+3; i++){
    for(int j = column*3; j < column*3+3; j++){
      for(int k = 0; k < 9; k++){
        if(sudoku[i][j] == compare_subgrids[k]){    //현재 원소와 비교용 배열과 비교
          compare_subgrids[k] = 0;                  //같으면 비교용 배열을 0 으로 치환
          subgrids_cnt++;                           //해주고 subgrids_cnt 값을 1 증가
        }
      }
    }
  }
  subgrids_num = row*3 + column;              //subgrids_num 을 통해 subgrid 번호 확인
  if(subgrids_cnt == 9) valid[2][subgrids_num] = 1;
    //subgrids_cnt 가 9 이면 valid[2][subgrids_num]를 1 로 바꿈
```

```c
  else valid[2][subgrids_num] = 0;                //9가 아니라면 0으로 바꿈
  pthread_exit(NULL);                             //프로세스 종료
}

/*
 * 스도쿠 퍼즐이 올바르게 구성되어 있는지 11개의 스레드를 생성하여 검증한다.
 * 한 스레드는 각 행이 올바른지 검사하고, 다른 한 스레드는 각 열이 올바른지 검사한다.
 * 9개의 3x3 서브그리드에 대한 검증은 9개의 스레드를 생성하여 동시에 검사한다.
 */
void check_sudoku(void)
{
    int i, j, num;
    pthread_t check[11];  //0번~8번 subgrid, 9번 row, 10번 column

    /*
     * 검증하기 전에 먼저 스도쿠 퍼즐의 값을 출력한다.
     */
    for (i = 0; i < 9; ++i) {
        for (j = 0; j < 9; ++j)
            printf("%2d", sudoku[i][j]);
        printf("\n");
    }
    printf("---\n");
    /*
     * 스레드를 생성하여 각 행을 검사하는 check_rows() 함수를 실행한다.
     */
    pthread_create(&check[9], NULL, check_rows, NULL);
    /*
     * 스레드를 생성하여 각 열을 검사하는 check_columns() 함수를 실행한다.
     */
    pthread_create(&check[10], NULL, check_columns, NULL);
    /*
     * 9개의 스레드를 생성하여 각 3x3 서브그리드를 검사하는 check_subgrid() 함수를 실행한다.
     * 3x3 서브그리드의 위치를 식별할 수 있는 값을 함수의 인자로 넘긴다.
     */
    for(i = 0; i < 3; i++){
      for(j = 0; j < 3; j++){
        //location_t을 동적할당해준다.
        location_t *data = (location_t *)malloc(sizeof(location_t));
        data -> row = i;                      //row와 column을 i, j에 대입
        data -> column = j;
        num = i*3 + j;                        //num을 이용해 0~8의 번호를 가진 thread생성
        if(pthread_create(&check[num],NULL,check_subgrid, data) != 0){
            fprintf(stderr, "pthread_create error: check_subgrid\n");
            exit(-1);
        }
      }
    }
    /*
     * 11개의 스레드가 종료할 때까지 기다린다.
     */
```

```
    for(i = 0; i <=11; i++){
      pthread_join(check[i], NULL);
    }
    /*
     * 각 행에 대한 검증 결과를 출력한다.
     */
    printf("ROWS: ");
    for (i = 0; i < 9; ++i)
        printf(valid[0][i] == 1 ? "(%d,YES)" : "(%d,NO)", i);
    printf("\n");
    /*
     * 각 열에 대한 검증 결과를 출력한다.
     */
    printf("COLS: ");
    for (i = 0; i < 9; ++i)
        printf(valid[1][i] == 1 ? "(%d,YES)" : "(%d,NO)", i);
    printf("\n");
    /*
     * 각 3x3 서브그리드에 대한 검증 결과를 출력한다.
     */
    printf("GRID: ");
    for (i = 0; i < 9; ++i)
        printf(valid[2][i] == 1 ? "(%d,YES)" : "(%d,NO)", i);
    printf("\n---\n");
}

/*
 * 스도쿠 퍼즐의 값을 3x3 서브그리드 내에서 무작위로 섞는 함수이다.
 */
void *shuffle_sudoku(void *arg)
{
    int i, tmp;
    int grid;
    int row1, row2;
    int col1, col2;

    srand(time(NULL));
    for (i = 0; i < 100; ++i) {
        /*
         * 0 부터 8 번 사이의 서브그리드 하나를 무작위로 선택한다.
         */
        grid = rand() % 9;
        /*
         * 해당 서브그리드의 좌측 상단 행열 좌표를 계산한다.
         */
        row1 = row2 = (grid/3)*3;
        col1 = col2 = (grid%3)*3;
        /*
         * 해당 서브그리드 내에 있는 임의의 두 위치를 무작위로 선택한다.
         */
        row1 += rand() % 3; col1 += rand() % 3;
```

```
        row2 += rand() % 3; col2 += rand() % 3;
        /*
         * 홀수 서브그리드이면 두 위치에 무작위 수로 채우고,
         */
        if (grid & 1) {
            sudoku[row1][col1] = rand() % 8 + 1;
            sudoku[row2][col2] = rand() % 8 + 1;
        }
        /*
         * 짝수 서브그리드이면 두 위치에 있는 값을 맞바꾼다.
         */
        else {
            tmp = sudoku[row1][col1];
            sudoku[row1][col1] = sudoku[row2][col2];
            sudoku[row2][col2] = tmp;
        }
    }
    pthread_exit(NULL);
}

/*
 * 메인 함수는 위에서 작성한 함수가 올바르게 동작하는지 검사하기 위한 것으로 수정하면 안 된다.
 */
int main(void)
{
    int tmp;
    pthread_t tid;

    /*
     * 기본 스도쿠 퍼즐을 출력하고 검증한다.
     */
    check_sudoku();
    /*
     * 기본 퍼즐에서 값 두개를 맞바꾸고 검증해본다.
     */
    tmp = sudoku[5][3]; sudoku[5][3] = sudoku[6][2]; sudoku[6][2] = tmp;
    check_sudoku();
    /*
     * 기본 스도쿠 퍼즐로 다시 바꾼 다음, shuffle_sudoku 스레드를 생성하여 퍼즐을 섞는다.
     */
    tmp = sudoku[5][3]; sudoku[5][3] = sudoku[6][2]; sudoku[6][2] = tmp;
    if (pthread_create(&tid, NULL, shuffle_sudoku, NULL) != 0) {
        fprintf(stderr, "pthread_create error: shuffle_sudoku\n");
        exit(-1);
    }
    /*
     * 무작위로 섞는 중인 스도쿠 퍼즐을 검증해본다.
     */
    check_sudoku();
    /*
     * shuffle_sudoku 스레드가 종료될 때까지 기다란다.
```

```
    */
    pthread_join(tid, NULL);
    /*
     * shuffle_sudoku 스레드 종료 후 다시 한 번 스도쿠 퍼즐을 검증해본다.
     */
    check_sudoku();
    exit(0);
}
```

프로그램 컴파일 과정

```
> gcc -v proj2-1.skeleton.c
Apple clang version 12.0.0 (clang-1200.0.32.27)
Target: x86_64-apple-darwin20.3.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
 "/Library/Developer/CommandLineTools/usr/bin/clang" -cc1 -triple x86_64-apple-macosx11.0.0 -Wdeprecated-objc-isa-usage -Werror=deprecated-objc-isa-usage -Werror=implicit-function
-declaration -emit-obj -mrelax-all -disable-free -disable-llvm-verifier -discard-value-names -main-file-name proj2-1.skeleton.c -mrelocation-model pic -pic-level 2 -mthread-model
posix -mframe-pointer=all -fno-strict-return -masm-verbose -munwind-tables -target-sdk-version=11.0 -target-cpu penryn -dwarf-column-info -debugger-tuning=lldb -target-linker-vers
ion 609.6 -v -resource-dir /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0 -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk -I/usr/local/include -internal-is
ystem /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/local/include -internal-isystem /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/include -internal-externc-is
ystem /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include -internal-externc-isystem /Library/Developer/CommandLineTools/usr/include -Wno-reorder-init-list -Wno-implici
t-int-float-conversion -Wno-c99-designator -Wno-final-dtor-non-final-class -Wno-extra-semi-stmt -Wno-misleading-indentation -Wno-quoted-include-in-framework-header -Wno-implicit-f
allthrough -Wno-enum-enum-conversion -Wno-enum-float-conversion -fdebug-compilation-dir /Users/lys/desktop/univ/cpp -ferror-limit 19 -fmessage-length 80 -stack-protector 1 -fstack
-check -mdarwin-stkchk-strong-link -fblocks -fencode-extended-block-signature -fregister-global-dtors-with-atexit -fgnuc-version=4.2.1 -fobjc-runtime=macosx-11.0.0 -fmax-type-alig
n=16 -fdiagnostics-show-option -fcolor-diagnostics -o /var/folders/4v/gvsn66p55kb82v8gyphfgp_r0000gn/T/proj2-1-d9fe20.o -x c proj2-1.skeleton.c
clang -cc1 version 12.0.0 (clang-1200.0.32.27) default target x86_64-apple-darwin20.3.0
ignoring nonexistent directory "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/local/include"
ignoring nonexistent directory "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/Library/Frameworks"
#include "..." search starts here:
#include <...> search starts here:
 /usr/local/include
 /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/include
 /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include
 /Library/Developer/CommandLineTools/usr/include
 /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/System/Library/Frameworks (framework directory)
End of search list.
 "/Library/Developer/CommandLineTools/usr/bin/ld" -demangle -lto_library /Library/Developer/CommandLineTools/usr/lib/libLTO.dylib -no_deduplicate -dynamic -arch x86_64 -platform_v
ersion macos 11.0.0 11.0 -syslibroot /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk -o a.out -L/usr/local/lib /var/folders/4v/gvsn66p55kb82v8gyphfgp_r0000gn/T/proj2-1-d9fe20.
o -lSystem /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/lib/darwin/libclang_rt.osx.a
```

실행 결과물

1번은 알맞은 스도쿠를 입력하여 맞는지 확인한 것이기 때문에 모든 값들이 YES를 출력한다. 또한 주어지는 스도쿠가 항상 같기때문에 같은 결과를 출력한다.

```
> ./a.out
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 4 6 9 5 2 1
 5 4 2 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
COLS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
GRID: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
---
```

2번은 1번과 같은 스도쿠에서 두개의 숫자를 맞바꾸고 맞는지 확인한 것이다. 두개의 스도쿠를 바꾸었기 때문에 그 두개가 해당하는 row, column, grid가 NO가 출력되게 된다.

```
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 2 6 9 5 2 1
 5 4 4 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,NO)(6,NO)(7,YES)(8,YES)
COLS: (0,YES)(1,YES)(2,NO)(3,NO)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
GRID: (0,YES)(1,YES)(2,YES)(3,YES)(4,NO)(5,YES)(6,NO)(7,YES)(8,YES)
---
```

3 번은 무작위로 섞는 중인 스도쿠를 검사하는 것으로 출력되는 스도쿠와 check_sudoku는 일치하지 않는다. 또한 time값을 seed로 갖고 있기 때문에 매번 다른 결과가 나오게된다.

```
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 4 6 9 5 2 1
 5 4 2 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
COLS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
GRID: (0,YES)(1,NO)(2,YES)(3,NO)(4,YES)(5,NO)(6,YES)(7,NO)(8,YES)
---
```

4번은 shuffle이 끝난 스도쿠를 검사하는 것으로 출력되는 스도쿠와 check_sudoku의 결과는 일치한다.

```
 8 9 2 4 2 7 4 6 2
 4 6 7 5 4 4 1 7 5
 3 5 1 7 1 8 9 8 3
 3 2 7 7 5 4 1 8 3
 8 2 1 3 2 1 3 3 5
 3 8 1 6 8 9 7 4 5
 5 1 6 1 1 5 9 1 3
 3 7 2 3 1 8 8 5 6
 9 8 4 8 1 8 2 4 7
---
ROWS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
COLS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
GRID: (0,YES)(1,NO)(2,YES)(3,NO)(4,YES)(5,NO)(6,YES)(7,NO)(8,YES)
---
```

한번 더 실행했을 때의 결과

```
> ./a.out
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 4 6 9 5 2 1
 5 4 2 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
COLS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
GRID: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
---
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 2 6 9 5 2 1
 5 4 4 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0,YES)(1,YES)(2,YES)(3,YES)(4,YES)(5,NO)(6,NO)(7,YES)(8,YES)
COLS: (0,YES)(1,YES)(2,NO)(3,NO)(4,YES)(5,YES)(6,YES)(7,YES)(8,YES)
GRID: (0,YES)(1,YES)(2,YES)(3,YES)(4,NO)(5,YES)(6,NO)(7,YES)(8,YES)
---
```

```
 6 3 9 8 4 1 2 7 5
 7 2 4 9 5 3 1 6 8
 1 8 5 7 2 6 3 9 4
 2 5 6 1 3 7 4 8 9
 4 9 1 5 8 2 6 3 7
 8 7 3 4 6 9 5 2 1
 5 4 2 3 9 8 7 1 6
 3 1 8 6 7 5 9 4 2
 9 6 7 2 1 4 8 5 3
---
ROWS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
COLS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
GRID: (0,YES)(1,NO)(2,YES)(3,NO)(4,YES)(5,NO)(6,YES)(7,NO)(8,YES)
---
 9 7 3 4 4 4 3 4 6
 8 1 2 9 5 5 5 8 2
 5 6 4 4 5 1 7 1 9
 5 4 7 1 3 9 1 2 2
 7 3 5 6 8 5 6 6 7
 8 4 2 4 7 2 5 7 3
 4 7 1 4 2 1 7 1 5
 6 3 9 5 6 2 6 3 4
 8 5 2 7 1 2 2 8 9
---
ROWS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
COLS: (0,NO)(1,NO)(2,NO)(3,NO)(4,NO)(5,NO)(6,NO)(7,NO)(8,NO)
GRID: (0,YES)(1,NO)(2,YES)(3,NO)(4,YES)(5,NO)(6,YES)(7,NO)(8,YES)
---
```