Backtracking

Stop it, before it’s too late !

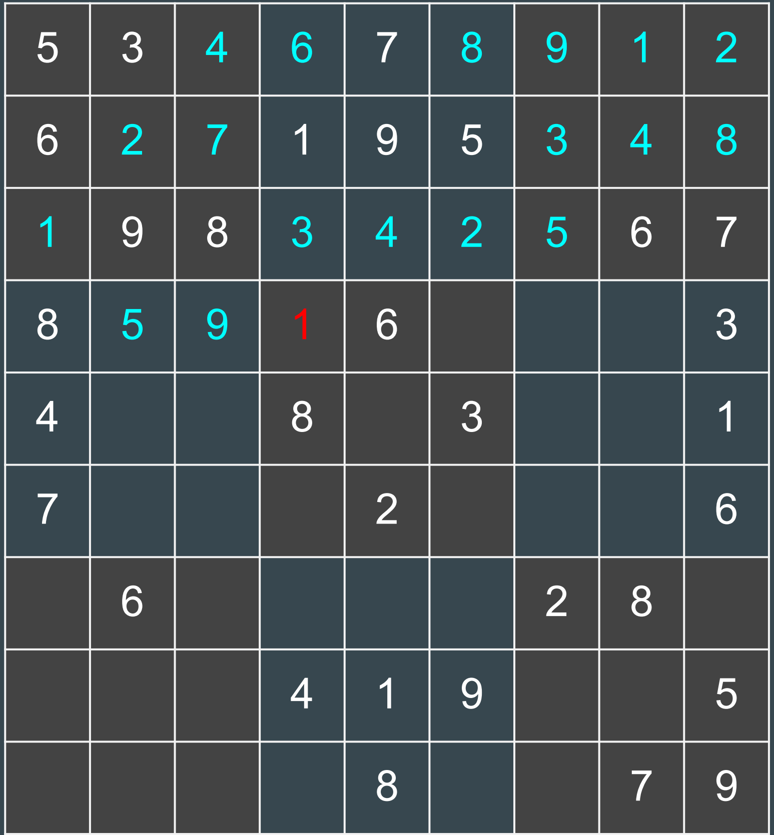
Backtracking - introduction

We need to reduce the *search space*

Example: sudoku

- Try to fill the sudoku
- Everytime you encounter a conflict, change the last number
- If no number is ok, erase and change the previous one

etc



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

Backtracking - concepts

The idea of backtracking is to:

- build the elements of the search space incrementally
- eliminate wrong partial solutions → and therefore all solutions that contain them

You can think of the search space as a *tree*, you will often use a *recursive function*

E.g for sudoku, each level of the tree corresponds to a empty cell in the grid

→ the size of the search space is 9^n but many cases can be discarded

Implementation of a backtracking sudoku solver in Python

```
def is_valid(grid, i, j, val):
    line = grid[i]
    column = [grid[k][j] for k in range(9)]
    square = [grid[3 * (i // 3) + k][3 * (j // 3) + l]
              for k in range(3) for l in range(3)]
    return not (val in line or val in column or val in square)

def backtracking(grid, i, j):
    if i == 9: return True
    nexti, nextj = (i if j < 8 else i + 1), (j + 1) % 9
    if grid[i][j] != 0:
        return backtracking(grid, nexti, nextj)
    for val in range(1, 10):
        if is_valid(grid, i, j, val):
            grid[i][j] = val
            if backtracking(grid, nexti, nextj): return True
            grid[i][j] = 0
    return False
```

```
grid = [list(map(int, input().split()))
        for _ in range(9)]

if not backtracking(grid, 0, 0):
    print("Impossible")
else:
    print("\n".join(" ".join(
            map(str, grid[i]))
            for i in range(9)))
```

Implementation of a backtracking sudoku solver in Python

```
IN:

5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

```
OUT:

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

Partial candidates explored: 6428

Total size of the workspace: $8{,}86293812\times10^{21}$

Credits

Slides:	Louis Sugy, Arthur Tondereau
Sudoku sample:	Wikipedia