# گزارش تمرین عملی HW3:

در بخش اول برای حل سودوکو از روش backtracking ساده استفاده کردیم. تابع اصلی حل مسئله در pdf موجود بود و برای تکمیل کد تابع های زیر پیاده سازی شد:

1. getNextLocation: در این تابع در یک حلقه دو بعدی تمام خانه های جدول بررسی شده و اولین خانه خالی برگردانده میشود. در صورت پر بودن همه خانه ها 1- برگردانده میشود.
2. isSafe: در این تابع بررسی میشود که مقدار انتخابی برای یک خانه با خانه های اطراف و constrait های مسئله تناقض دارد یا خیر.

در بخش دوم برای حل سودوکو از روش csp به همراه ordering و forwardChecking استفاده شده. این روش مشابه همان backtracking است با این تفاوت که از یک آرایه دو بعدی به نام RV برای ذخیره کردن دامنه خانه های سودوکو استفاده شده.

در این بخش از کد موجود در pdf برای تعیین مقدار دامنه ها استفاده شد. از کد solveSimpleBacktracking به عنوان هسته اصلی راه حل استفاده شد اما با ایجاد برخی تغییرات، سرعت اجرا تا 1000 برابر بهبود پیدا کرد که نتایج آن بررسی خواهد شد. در بخش دوم توابع زیر به کد اضافه شدند:

1. forwardChecking: این تابع یک لیست از خانه هایی که دامنه آنها تغییر کرده دریافت میکند و اگر دامنه یک نود خالی شده باشد مقدار False میگرداند تا تابع بازگردد و راه اشتباه را ادامه ندهد.
2. getNextLocation: این تابع از میان خانه های جدول خانه ای که کم ترین دامنه را دارد انتخاب میکند.

3. removeFromDomain: وقتی به یک خانه مقداری اساین میشود این تابع آن مقدار را از دامنه خانه های همسایه حذف میکند.
4. recoverDomain: در صورتی که مقدار اشتباهی به یک خانه اساین شده باشد، این تابع دامنه تغییر کرده خانه های همسایه را به حالت اول بر میگرداند.
5. Csp: این تابع الگوریتم اصلی را اجرا میکند.

## تفاوت الگوریتم بخش اول (solveSimpleBacktracking) و بخش دوم( csp with ordering and forward checking)

تفاوت های این دو الگوریتم به شرح زیر است:

1. تفاوت این دو الگوریتم این است که در بخش اول، getNextLocation اولین خانه خالی را بر میگرداند اما در بخش دوم تمام دامنه خانه ها در آرایه RV ذخیره شده و خانه ای که کوچک ترین دامنه را دارد برگردانده میشود.
2. در روش اول از forwardChecking استفاده نمیشود اما در روش دوم استفاده میشود.

## تحلیل زمان اجرا و expandedNode های دو روش:

در روش دوم، تابع اصلاح شده getNextLocation و تابع forwardChecking توانسته تعداد نود های گسترش یافته را بسیار زیاد کاهش دهد و در مراحل کم تری به پاسخ برسد. همچنین با استفاده از توابع نوشته شده removeFromDomain و recoverDomain ، آرایه RV با ایجاد هر تغییر در خانه ها آپدیت میشود و لازم نیست هر بار در ابتدای تابع csp، تمام خانه ها بررسی شوند و دامنه آن ها دوباره حساب شود که باعث میشود در زمان بسیار صرفه جویی شود.

برای بررسی راحت تر از سه نوع جدول سودوکو با سختی های آسان و متوسط و سخت استفاده شده تا تغییرات دو روش ملموس تر باشد.

1. جدول اول – جدول موجود در pfd:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 6 | 3 | 0 | 8 | 4 | 2 | 0 |
| 8 | 4 | 0 | 0 | 0 | 7 | 3 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 6 | 0 | 9 | 4 | 0 | 8 | 0 | 2 |
| 0 | 8 | 1 | 0 | 3 | 0 | 7 | 9 | 0 |
| 9 | 0 | 3 | 0 | 7 | 6 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 5 | 7 | 0 | 0 | 0 | 6 | 8 |
| 0 | 7 | 8 | 1 | 0 | 3 | 2 | 5 | 0 |

نتایج اجرا:

```
Test1
----------------------------------------
backtracking
True
['7', '1', '6', '3', '5', '8', '4', '2', '9']
['8', '4', '9', '2', '6', '7', '3', '1', '5']
['3', '5', '2', '4', '1', '9', '6', '8', '7']
['5', '6', '7', '9', '4', '1', '8', '3', '2']
['4', '8', '1', '5', '3', '2', '7', '9', '6']
['9', '2', '3', '8', '7', '6', '5', '4', '1']
['2', '9', '4', '6', '8', '5', '1', '7', '3']
['1', '3', '5', '7', '2', '4', '9', '6', '8']
['6', '7', '8', '1', '9', '3', '2', '5', '4']
expanded nodes: 371
time: 18 ms

csp with ordering
True
['7', '1', '6', '3', '5', '8', '4', '2', '9']
['8', '4', '9', '2', '6', '7', '3', '1', '5']
['3', '5', '2', '4', '1', '9', '6', '8', '7']
['5', '6', '7', '9', '4', '1', '8', '3', '2']
['4', '8', '1', '5', '3', '2', '7', '9', '6']
['9', '2', '3', '8', '7', '6', '5', '4', '1']
['2', '9', '4', '6', '8', '5', '1', '7', '3']
['1', '3', '5', '7', '2', '4', '9', '6', '8']
['6', '7', '8', '1', '9', '3', '2', '5', '4']
expand nodes: 44
time: 1 ms
```

2. جدول دوم:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | 4 | 0 | 0 | 0 | 0 | 0 | 6 | 2 |
| 0 | 2 | 0 | 9 | 0 | 0 | 8 | 0 | 5 |
| 0 | 0 | 0 | 0 | 2 | 5 | 0 | 0 | 7 |
| 4 | 0 | 0 | 2 | 1 | 0 | 0 | 5 | 0 |
| 0 | 0 | 0 | 0 | 0 | 9 | 7 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 |
| 0 | 1 | 0 | 7 | 4 | 0 | 5 | 0 | 0 |
| 2 | 0 | 0 | 3 | 0 | 8 | 0 | 0 | 0 |
| 6 | 8 | 0 | 5 | 0 | 0 | 3 | 0 | 0 |

نتایج اجرا:

```
Test2
----------------------------------------
backtracking
True
['3', '4', '5', '1', '8', '7', '9', '6', '2']
['7', '2', '6', '9', '3', '4', '8', '1', '5']
['8', '9', '1', '6', '2', '5', '4', '3', '7']
['4', '7', '9', '2', '1', '3', '6', '5', '8']
['1', '6', '2', '8', '5', '9', '7', '4', '3']
['5', '3', '8', '4', '7', '6', '2', '9', '1']
['9', '1', '3', '7', '4', '2', '5', '8', '6']
['2', '5', '4', '3', '6', '8', '1', '7', '9']
['6', '8', '7', '5', '9', '1', '3', '2', '4']
expanded nodes: 968
time: 21 ms

csp with ordering
True
['3', '4', '5', '1', '8', '7', '9', '6', '2']
['7', '2', '6', '9', '3', '4', '8', '1', '5']
['8', '9', '1', '6', '2', '5', '4', '3', '7']
['4', '7', '9', '2', '1', '3', '6', '5', '8']
['1', '6', '2', '8', '5', '9', '7', '4', '3']
['5', '3', '8', '4', '7', '6', '2', '9', '1']
['9', '1', '3', '7', '4', '2', '5', '8', '6']
['2', '5', '4', '3', '6', '8', '1', '7', '9']
['6', '8', '7', '5', '9', '1', '3', '2', '4']
expand nodes: 51
time: 2 ms
```

3. جدول سوم:

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

نتایج اجرا:

```
Test3
--------------------------------------
backtracking
True
['6', '7', '8', '1', '4', '3', '9', '5', '2']
['2', '4', '9', '7', '5', '6', '3', '1', '8']
['5', '3', '1', '8', '2', '9', '7', '4', '6']
['3', '2', '6', '4', '7', '8', '1', '9', '5']
['7', '1', '5', '6', '9', '2', '8', '3', '4']
['9', '8', '4', '5', '3', '1', '6', '2', '7']
['8', '5', '3', '9', '6', '4', '2', '7', '1']
['4', '6', '2', '3', '1', '7', '5', '8', '9']
['1', '9', '7', '2', '8', '5', '4', '6', '3']
expanded nodes: 179970
time: 3269 ms

csp with ordering
True
['6', '7', '8', '1', '4', '3', '9', '5', '2']
['2', '4', '9', '7', '5', '6', '3', '1', '8']
['5', '3', '1', '8', '2', '9', '7', '4', '6']
['3', '2', '6', '4', '7', '8', '1', '9', '5']
['7', '1', '5', '6', '9', '2', '8', '3', '4']
['9', '8', '4', '5', '3', '1', '6', '2', '7']
['8', '5', '3', '9', '6', '4', '2', '7', '1']
['4', '6', '2', '3', '1', '7', '5', '8', '9']
['1', '9', '7', '2', '8', '5', '4', '6', '3']
expand nodes: 117
time: 4 ms
```

همانطور که مشاهده میشود تعداد نود های گسترش یافته و زمان اجرا تا 1000 برابر در روش دوم بهتر شده.