توضیحی بر الگوریتم حل سودوکو

الگوریتم های مختلفی برای حل سودوکو وجود دارد که ما یکی از آنها را به طور خلاصه توضیح می دهیم:

نوع داده ای که برای جدول سودوکو استفاده میکنیم لیست دوبعدی است. یعنی در ورودی یک لیست دو بعدی تعریف می شود که در آن خانه هایی که از قبل مقدارشان مشخص است و خانه های خالی که با صفر مشخص شده اند، وجود دارند.

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

به عنوان مثال سودوکو بالا را به شکل لیست دو بعدی زیر نشان می دهیم:

```
[[3, 0, 6, 5, 0, 8, 4, 0, 0],
[5, 2, 0, 0, 0, 0, 0, 0, 0],
[0, 8, 7, 0, 0, 0, 0, 3, 1],
[0, 0, 3, 0, 1, 0, 0, 8, 0],
[9, 0, 0, 8, 6, 3, 0, 0, 5],
[0, 5, 0, 0, 9, 0, 6, 0, 0],
[1, 3, 0, 0, 0, 0, 2, 5, 0],
[0, 0, 0, 0, 0, 0, 0, 7, 4],
[0, 0, 5, 2, 0, 6, 3, 0, 0]]
```

الگوریتم کلی به این صورت است که ما میتوانیم سودوکو را با جایگذاری اعداد در خانه های خالی حل کنیم.

قبل از اینکه بخواهیم عددی را جایگذاری کنیم باید چک کنیم که اضافه کردن آن عدد قوانین بازی سودوکو را نقض نکند. یعنی چک می کنیم که عددی که میخواهیم اضافه کنیم در همان سطر و ستون و همچنین مربع ۳*۳ شامل آن وجود نداشته باشد. اگر عدد مورد نظر شرایطی که گفتیم را داشت آن عدد را اضافه میکنیم و سپس به طور بازگشتی چک میکنیم عددی که اضافه کردیم ما را می تواند به جواب برساند یا خیر. یعنی میتوانیم جاهای خالی دیگر را نیز با فرض اینکه این عدد خاص اضافه شده است پر کنیم.

اگر این عدد ما را نتوانست به جوابی برساند عددی دیگر را برای آن جای خالی تست میکنیم و به همین ترتیب ادامه میدهیم.

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

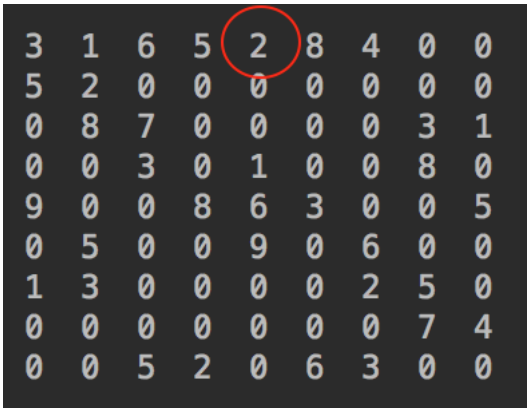
مثلا برای همین شکل بالا ابتدا اولین جای خالی را پیدا میکنیم. درشکل اولین جای خالی با دایره قرمز مشخص شده است.

سپس از اعداد ۱ تا ۹ باید یک عدد را در دایره قرمز که مشخص شده قرار دهیم. از عدد ۱ شروع میکنیم هر عددی که قوانین سودوکو را نقض نکرد جایگذاری میکنیم. در این مثال ما ابتدا عدد ۱ را جایگذاری میکنیم و میبینیم که همین عدد ۱ هیچ یک از قوانین را نقض نمیکند. بعد از جایگذاری عدد ۱ جدول به صورت زیر در می آید:

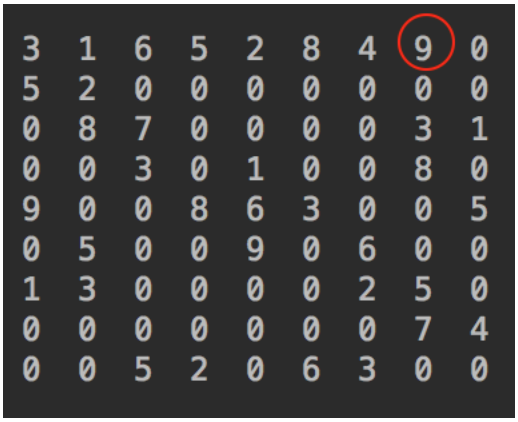
3	1	6	5	0	8	4	0	0
5	2	0	0	0	0	0	0	0
0	8	7	0	0	0	0	3	1
0	0	3	0	1	0	0	8	0
9	0	0	8	6	3	0	0	5
0	5	0	0	9	0	6	0	0
1	3	0	0	0	0	2	5	0
0	0	0	0	0	0	0	7	4
0	0	5	2	0	6	3	0	0

حال باید به طور بازگشتی همان کار را دوباره انجام دهیم. یعنی اولین صفر را پیدا کنیم و عدد مناسب را در صورت امکان در آن قرار دهیم.(پیدا کردن صفر ها را سطر به سطر انجام می دهیم.)

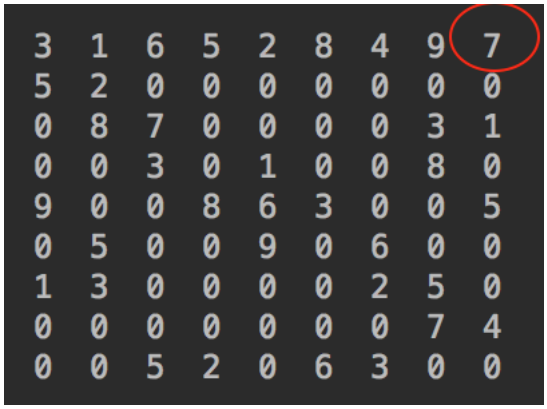
حالت بعدی به صورت زیر در می آید:



حالت بعدی:



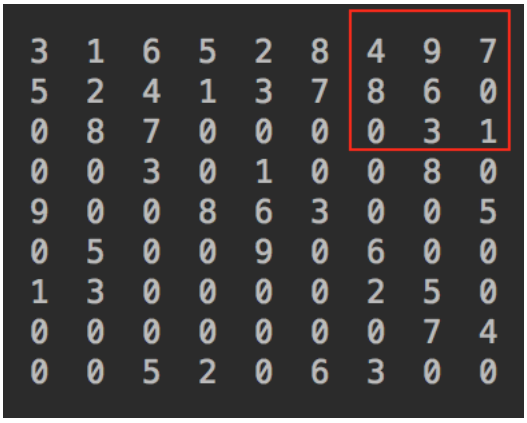
حالت بعدی:



همان طور که در حالت های بالا دیدید برنامه باید با استفاده از قوانین به ترتیب سطر ها را پر کند. اولین جای خالی را پیدا کند. عدد مناسبی در آن قرار دهد و دوباره به صورت بازگشتی جای خالی بعدی را پیدا کند و عدد مناسبی در آن قرار دهد.

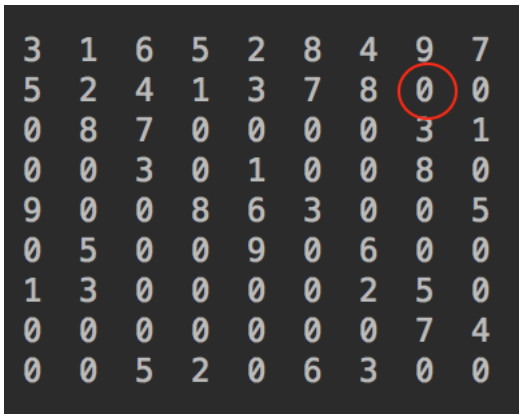
دقت کنید که ممکن است حالتی پیش بیاید که ما نتوانیم با توجه به اعدادی که قبلا جایگذاری کرده ایم در جای خالی بعدی عددی را قرار دهیم. در این حالت باید به اعدادی که قبلا مقداردهی کردیم برگردیم و مقدار دیگری را برای آن ها انتخاب کنیم.

برای مثال اگر الگوریتم را ادامه دهیم در جایی به شکل زیر میرسیم:



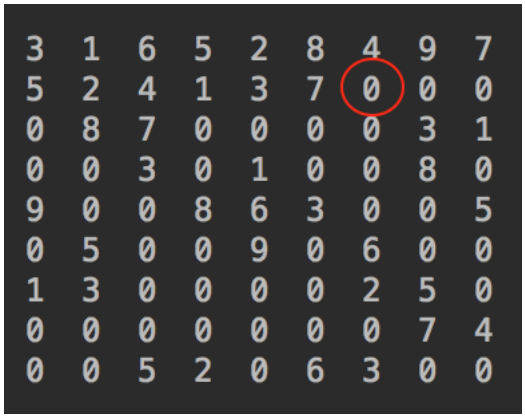
همانطور که میبینید سطر اول به طور کامل پر شده و سطر دوم عدد آخر آن باید پر شود. اما تنها عددی که در این سطر موجود نیست عدد ۹ است که به دلیل این که در مربع ۳*۳ عدد ۹ دیگری وجود دارد نمیتوان ۹ را به جای صفر سطر دوم قرار داد. پس باید به عقب بازگردیم و اعدادی که تا الان اضافه کرده بودیم را تغییر دهیم.

برای بازگشت به عقب باید دوباره عدد ۶ را به صفر تبدیل کنیم و عددی دیگر را جای آن بگذاریم. یعنی ابتدا جدول ما به شکل زیر در می آید



سپس سعی میکنیم عدد دیگری به جای ۶ انتخاب کنیم. اگر موفق بودیم جلو میرویم و اگر ناموفق بودیم دوباره بازگشت به عقب دیگری انجام میدهیم.

در این مثال به جای عدد ۶ هیچ عدد دیگری نمیتواند قرار بگیرد چون قوانین را نقض میکند و ما دوباره باید به عقب برگردیم یعنی ۸ را تبدیل به صفر کنیم. دوباره به شکل زیر میرسیم:



و به همین ترتیب بقیه ی کار را ادامه می دهیم.

فقط دقت داشته باشید که برنامه وقتی تمام میشود که هیچ عدد صفری در سودوکو باقی نماند و در واقع سودوکو حل شده باشد.