

**دانشکده: مهندسی کامپیوتر**

**موضوع: داک تمرین دوم AI**

**علی شکوهی**

**شماره دانشجویی: 400521477**

## مراحل ساختن جدول سودوکو

الگوریتم کلی به این صورت است که ابتدا یک گرید 9\*9 می‌سازیم تا صفحه سودوکو تشکیل شود. سپس با استفاده از توابع موجود در این بخش، به این صورت جدول را پر می‌کنیم که ابتدا جدول به صورت قطری از خانه بالا سمت چپ به صورت خانه‌های 3\*3 به سمت پایین پر می‌شود.(یعنی از خانه (0,0) به سمت (8,8) پر می‌شود.) که برای این کار نیاز به توابع fill\_diagonal و fill\_box و تابعی برای چک کردن این هر عدد داخل هر خانه است که خود این تابع نیاز به سه تابع used\_in\_row، used in\_column و used\_in\_box است که هرکدام چک می‌کنند که قوانین بازی نقض نشود.

پس از پر کردن هر باکس به صورت قطری، باید سایر خانه‌ها نیز پر شود. در این قسمت از تابع find\_unassigned\_location برای چک کردن این که کدام خانه‌ها خالی است، استفاده می‌شود. همچنین از تابع fill\_unassigned\_locations برای پر کردن خانه‌های خالی به صورت رندوم استفاده می‌کنیم و در هنگام پرکردن هرخانه، با استفاده از تابع is\_safe چک می‌کنیم که هیچ conflictای با سایر خانه‌ها وجود نداشته باشد.

در آخر پس از پرکردن همه خانه‌ها، با استفاده از تابع remove\_elements مقدار رندومی از خانه‌ها را خالی می‌کنیم تا جدول سودوکو تشکیل شود.

## حل جدول سودوکو

برای حل جدول با استفاده از روش backtracking، در هر مرحله چک می‌کنیم که خانه مد نظر خالی است یا پر. در صورت پر بودن خانه، True را برمی‌گردانیم و در غیر این صورت به آن خانه یک عدد رندوم assign می‌کنیم و چک می‌کنیم که با سایر خانه‌ها conflict دارد یا خیر که اگر مشکلی نداشت، آن عدد را در آن خانه قرار می‌دهیم. و در تابع display\_grid، سودوکوی حل شده را چاپ می‌کنیم و همچنین تعداد مراحل حل نیز چاپ می‌شود.

برای حل جدول با استفاده از روش CSP، ابتدا برای هر خانه خالی جدول دامنه مربوط به آن را به دست می‌آوریم و با استفاده از تابع is\_valid\_assignment، چک می‌کنیم که assign کردن مقداری به هر خانه معتبر است یا خیر. در تابع find\_unassigned\_location، خانه‌های خالی جدول را پیدا می‌کنیم تا آن‌ها را مقداردهی کنیم. همچنین در تابع solve\_csp به صورت بازگشتی به هر خانه از دامنه آن خانه مقداردهی می‌کنیم و اگر آن مقداردهی صحیح بود که به سراغ خانه‌های بعدی می‌رویم و در غیر اینصورت، دوباره به عقب برمی‌گردیم و آن را با مقدار دیگر مقداردهی می‌کنیم. همچنین این تابع در هر مرحله چک می‌کند که جدول حل شده است تا grid را ریترن کند. همچنین تعداد مراحل حل نیز چاپ می‌شود.