به نام خدا

# فرزان رحمانی – 99521271

## نتایج و گزارش الگوریتم ها

## مقایسه و گزارش نتایج:

با تست روی جداول مختلف به این نتایج رسیدیم:  
در الگوریتم CSP نسبت به backtracking تعداد گره های expand شده به طرز چشم گیری کاهش پیدا کرد. در بیشتر تست ها CSP سریع تر عمل کرد (زمان کمتر) ولی در برخی تست ها مدت زمان backtracking بهتر بود. این به دلیل overhead محاسبات forward checking و آنالیز است. همچنین search space مسئله هم کاهش پیدا کرد. در مسائل کوچکتر و معمول تر forward check بهتر عمل می کند ولی در مسائل بسیار پیچیده overhead بیشتر خودش را نشان می دهد.

# راهکارهای بهبود:

برای بهتر کردن overhead of forward checking بهتر است بجای چک کردن کل جدول فقط سطر، ستون و خانه سه در سه را چک کنیم تا سریع تر عمل کند. به علاوه، می توان از heuristic functions هایی استفاده کرد که حدس بزنند کدام مقدار برای یک دامنه مشخص متغیر بهتر است. مثلا این تابع می تواند مقداری را انتخاب کند که کمتر در جدول تکرار شده است. اگه متد های ما consistent باشند نتایج را بهتر می کنند.

## خلاصه:

اگرچه الگوریتم‌های عقبگرد تمایل دارند هر مشکلی را در تئوری حل کنند، اما به فضای زیادی از حافظه نیاز دارند و زمان زیادی را صرف می‌کنند. الگوریتم‌های CSP به منظور کوچک کردن فضای بزرگ و تقویت این الگوریتم‌ها معرفی شدند. با الگوریتم های Forward Checking خوب و توابع اکتشافی سازگار(consistent heuristic functions)، حل مسئله با سرعت بالا با نیاز به حافظه کم امکان پذیر خواهد بود.

در زیر خروجی کد من برای یک جدول نمونه را مشاهده میکنید:

جدول نمونه:  
['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']

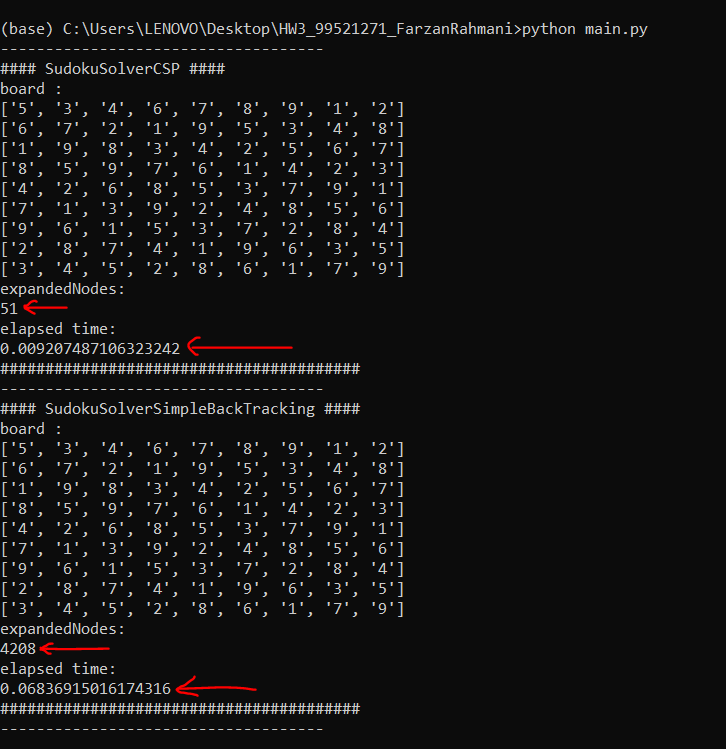
['4', '0', '0', '8', '0', '3', '0', '0', '1']

['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']



پایان