بسم الله الرحمن الرحیم

علی طاهری-40012323

*__نکته بسیار مهم:__ همه تایم های نوشته شده در این گزارش شامل پیدا کردن پاسخ + رسم خود جدول آن بصورت گرافیکی هست(که خود آن زمان بسیاری میگیرد)

در گزارش این تمرین ابتدا به توضیح مختصری راجب هر سوال میپردازیم:

__کد سوال اول:__ در این کد ما از روش جستجوی عقب گرد استفاده کرده، که در هر ستون یک وزیر را قرار میدهیم(برای هر ستون، ۹ حالت وجود دارد چون ۹ سطر داریم و هر وزیر را میتوان در یکی از این سطر ها قرار داد) برای یک ستون بعد از قرار دادن وزیر در یک سطر خاص(که سطر خاص از ۰ تا ۸ میتواند باشد)، محدودیت ها را چک میکنیم، که در انجا ایا جدول ما ایمن هست یا خیر، اگر ایمن بود به ستون بعدی میرویم و اگر ایمن نبود و محدودیت ها را براورده نمیکند، سطرهای بعدی را امتحان کرده(در همان ستون) و اگر در هیچ سطری نتوانستیم قرار دهیم به عقب بازگشته. این کار را ادامه میدهیم تا اینکه در همه ستون ها وزیری قرار گیرد و پاسخ را بدست اوریم.

چیزی که من در این روش مشاهده کردم این بود که اگر تعداد وزیر ها را از حدی بالاتر ببریم(مثلا اگر برای ۲۸ وزیر یا بیشتر امتحان کنیم) در زمانی معقول به جواب نخواهیم رسید(کمتر از یک دقیقه) و این الگوریتم برای تعداد بالای وزیر مناسب نیست.

کد سوال دوم: در این کد ما از روش جستجوی محلی استفاده کرده، بطوری که ابتدا در هر ستون، یک وزیر قرار میدهیم(سطر انرا بطور رندوم مشخص میکنیم) یک تابعی فراخوانی میکنیم، که در ان تابع ما یک ستون از ستون هایی که وزیرقرار داده شده در ان ایمن نیست و محدودیت ها در ان رعایت نشده است را بطور رندوم انتخاب میکنیم، و در ان ستون بررسی میکنیم کدام سطر است که اگر وزیر در ان سطر قرار گیرد کمترین تعداد تهدید را دارد و وزیر را در همان سطر قرار میدهیم(البته بجز سطر فعلی خودش) و این کار را تا انجایی ادامه میدهیم که یا هیچ دو وزیری همدیگر را تهدید نکنند و جدول ما ایمن باشد یا اینکه ما یک تعداد خاصی تکرار انجام دهیم.

در این کد مشاهده شد مسئله ۹ وزیر را تقریبا همیشه بطور صحیح حل میکند(از بین ۳۰۰ بار اجرا فقط یکبار به جواب نرسید) و از دقت بسیار بالایی برخورد دار است در مسئله ۹ وزیر.

همچنین مشاهده شد که تا مسئله ۱۴۰ وزیر را میتوان در تایمی در حدود ۴۰ ثانیه حل کرد(واضح است که اگر عمق بسیار زیاد باشد، شاید مجبور شویم چندبار الگوریتم را اجرا کنیم تا به جواب برسیم و باید حد تعداد تکرار را کمی بزرگتر در نظر بگیریم)

کد سوال سوم: در این کد که مشابه کد اول است، اما اضافه بر ان، ما از دو مکاشفه متغیر محدود کن بیشینه و مقدار محدود کن کمینه استفاده کرده ایم.

متغیرمحدود شده بیشینه: در این کد ابتدا ما مشخص میکنیم اگر الگوریتم تا یکجایی اجرا شده بود و میخواستیم بقیه وزیر ها را بچینیم، ان ستونی را انتخاب میکنیم و به سراغ ان ستونی میرویم

که کمترین مقدار ایمن برای ان هست و کمترین تعداد ایمن بودن را دارد(یعنی اگر همه ستون هایی که در ان وزیری نیست را در نظر بگیریم، در بین اینها ان ستونی که اگر ما وزیر را در سطر های مختلف ان قرار میدهیم و انجا ایمن هست، کمترین تعداد باشد، یعنی اگر در ۳ ستون که ما هنوز وزیری برای ان نگذاشته ایم، مقدار مجاز (یعنی در چند سطر میتوانستیم وزیر را قرار دهیم در ان ستون بطوری که ایمن باشد) به ترتیب ۳ ۵ ۶ باشد، ما ستونی که مقدار ان ۳ هست را ابتدا انتخاب میکنیم) و برای اولین بار که هنوز ما وزیری قرار نداده ایم ستونی را انتخاب میکنیم که در ان، در مجموع برای هر خانه در ان ستون بیشترین تعداد همسایه باشد.

مقدار محدود کن کمینه: یعنی اینکه در ستون انتخاب شده، کدام سطر را انتخاب کنیم؟ باید ان سطری را انتخاب کنیم که وقتی وزیر در ان سطر قرار گرفت مقدار محدودیت کمینه شود(مثلا کمترین میزان تهدید در انجا باشد) و وزیر را در ان سطر قرار میدهیم

در این کد چیزی که بطور حدودی(نه خیلی دقیق) تست کردم این بود که تا مسئله ۱۰۰ وزیر را به راحتی در زیر۳۵ ثانیه حل میکند و بسیار عالی عمل میکند.(قطعا برای وزیر های بیشتری نیز در تایم مناسب قابل حل است)

مشاهدات کلی و مقایسه با تمرین دوم:

من کد ۱ و ۲ تمرین دو را برای تعداد بالاتر وزیر(مثلا ۵۰ وزیر) اجرا کردم(خود استاد گفتند حداقل با یکی از کدهای تمرین ۲ مقایسه کنم)، در کد اول تمرین دوم ۰ بار از بین ۲۰ بار(دقت صفر درصد) و کد دوم تمرین دو، از بین ۲۰ بار اجرا تنها ۳ بار جواب صحیح را باز گرداند(دقت ۱۵ درصد). در حالی که در مسئله ۹ وزیر در تمرین دو برای سوال ۱ و ۲ به ترتیب، دقت ۲۰ و ۴۵ درصد بود. و نشان میدهد که کد ۱ و ۲ تمرین دوم در تعداد بالای وزیر به خوبی عمل نمیکند و عملکرد ان در تعداد بالاتر وزیر کاهش چشم گیری دارد.

در کد اول همین تمرین(تمرین سوم) برای تعداد بالای وزیر نیز به خوبی عمل نمیکند، اما همیشه جوابی که برمیگرداند صحیح است(یعنی هیچگاه شکست برنمیگرداند، ممکن است یک پاسخ را خیلی دیر برگرداند ولی هیچگاه اتفاق نمیافتد که یک جستجوی عقبگرد شکست برگرداند مگر اینکه خودمان یک شرط خیلی خاصی (مثلا یک شمارنده) برای ان بگذاریم)

در کد دوم همین تمرین(تمرین سوم)میتواند تعداد بالای وزیر را هم حل کند در تایمی معقول(برخلاف کد سوال ۱ همین تمرین) ولی همانطور که مشخص است بخاطر محدود بودن max_step ، اگر به اندازه max_step جلو رفته بودیم و به پاسخ نرسیده بودیم، آنگاه باید شکست را برگردانیم.

کد سوم را نیز برای حل وزیران در تایم معقول، با تعداد بالا میتوانیم استفاده بکنیم(مثلا ۱۰۰ وزیر) و همیشه پاسخ صحیح را برمیگرداند(شاید کمی دیر، ولی هیچگاه شکست نیست)