تمرین عملی 2: سودوکو با کمک از جستجو محلی

برای این تمرین، به شما یک جدول ناقص سودوکو داده میشود و شما باید جدول کامل آن را تحویل دهید.

نکات سریع:

- داده ورودی به صورت یک json هست که در آن 0 نشانه خالی بودن هست. مانند زیر:

```
{
"sudoku”: [
    [1,0,4, 8,6,5, 2,3,7],
    [7,0,5, 4,1,2, 9,6,8],
    [8,0,2, 3,9,7, 1,4,5],

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

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

- خروجی شما یک json مانند بالا هست با این تفاوت که مقادیر 0 جایگزین با مقدار صحیح شدند.
- سرعت، مصرف رم و تعداد سلول های غلط بر روی نمره تاثیر میگذارد.
- از استارتر کیت جدید برای این تمرین استفاده کنید. اسم کلاس AI و تابع solve تغییر نکند. بجز آن آزادید هرگونه کاری میخواهید انجام دهید.
- استفاده از git اجباری هست.
- تلاش میشود در آینده نزدیک یک نوع برنامه برای تست نیز به شما داده شود تا بتوانید برنامه خود را آزمایش کنید.
- هر گونه تقلب یا شباهت غیر قابل توجیه باعث کسر نمره میشود.

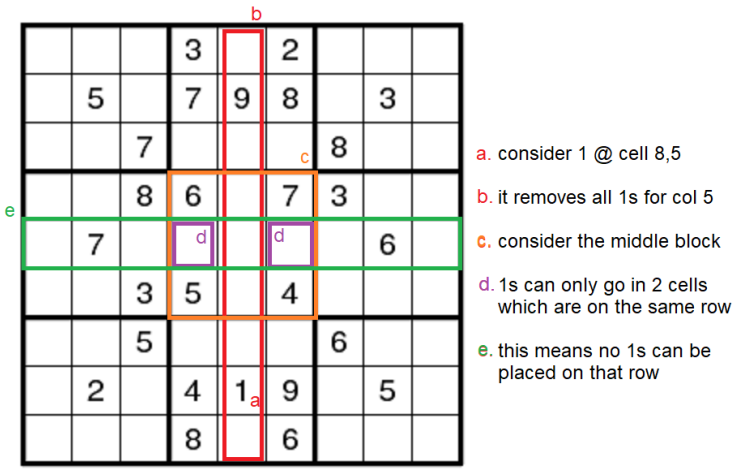
توضیح مسئله:

اگر با بازی سودوکو آشنا نیستید، به صورت کلی باید تلاش کنید که برای هر 9 سطر، 9 ستون و 9 بلوک، مقدار تکراری و یا خالی نداشته باشید. مقادیر ممکن بین 0 و 9 هستند. برای نمونه در سلول هایلایت شده، مقادیر 9،8،7،6،4،3،2،1 نمیتواند قرار گیرد (که یعنی فقط 5 ممکن هست):

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

تلاش برای حل مستقیم این مسئله برای سودوکو های دشوار میتواند چالش برانگیز باشد زیرا که باید الگو و روش هایی را بلد بود و همچنین به صورت کد پیاده کرد. **شما با این الگو ها کاریی ندارید و مسئله را به روش دیگری حل میکنید.**

فقط برای نمونه، یک الگوی ساده که برای بسیاری از سودوکو ها باید استفاده شود تا بتوان مقدار ناممکن را حذف کرد در پایین قرار دادیم:



حال تصور کنید که ده ها الگو مانند این وجود دارند که شما باید اول پیدا کرده و سپس پیاده سازی کنید. اما با روش هایی مثل جستجو محلی، میتوانید جواب را فقط با پیدا سازی قوانین بازی بدست می آورید.

راهنمایی :

با توجه به اینکه سرعت و مصرف حافظه در عملکرد و نیز ارزشیابی!!! مهم هستند چند نکته را در نظر داشته باشید.

کد الگوریتم های جست و جوی محلی مانند سایر الگوریتم ها به صورت آماده موجودند اما چیزی که موجب تفاوت در عملکرد عامل حل مسئله میشود، فورموله سازی مبتکرانه و نیز شهود شما درانتخاب اجزای الگوریتم هاست. برای مثال اگر از الگوریتم ژنتیک استفاده میکنید نوع تعریف کروموزوم ها، نحوه برش و ترکیب، نحوه انتخاب نسل بعد و... همه و همه با توجه به مسئله میتوانند در عملکرد الگوریتم موثر باشند. یا اگر از الگوریتم شبیه سازی گداخت استفاده میکنید متد های مختلفی برای سردکردن وجود دارند که میتوانید منابع مختلفی را در این مورد جست و جو کنید.

با توجه به همین انتظار میرود بخش قابل توجهی از کار شما کاوش و جستجو برای یافتن بهترین متد ها باشد.

موفق باشید.

نکته اضافه بر بحث:

چارچوب شبیه‌ساز محیط و تابع act در جستجوهای محلی در واقع جایگاه دیگری دارند. شکل "تعاملی" برای حل، در واقع نگرش درستی برای جستجوی محلی نیست. جستجوی محلی در کل، یک الگوریتم به معنای کلاسیک (غیرتعاملی) است و جواب نهایی را به ما میدهد.

اگر بخواهید در چارچوب تعاملی تعریف کنید میتوانید به صورت تغییر یک سودوکو غلط به آن فکر کنید. یعنی برای مثال برای حل سودوکو به صورت تعاملی، سودوکو را کاملا پر شده اما غلط ادراک میکنیدکند و هر act بمعنای پاک کردن محتوای یک خانه و نوشتن یک عدد دیگر در آن خانه باشد (مداد و مداد پاک‌کن). به زبان دیگر هر act میشود دادن آدرس یک خانه و یک عدد که بجای عدد قبلی نوشته می‌شود. (جاهای مشخصی از جدول ، البته غیرقابل تغییر هستند که همانا کلیدهای طرح هر سودکو هستند.) همچنین میشود تابع act خروجی به صورت batch دهد. مثلاً دو (یا بیشتر) عمل با هم انجام شوند. دو عمل باهم، تعبیر جالب‌تری دارد بویژه اگر شماره سطر یا شماره ستون آن دو خانه‌ای که می‌خواهد تغییرشان دهد یکسان باشد.

این ها فقط ایده های حل تعاملی هستند. الگ. های جستجو محلی معمولا جواب نهایی را میدهند و به همین دلیل اسم تابع به solve تغییر کرده.