مسئله هشت وزیر

در این پروژه مسئله معروف قرار دادن هشت وزیر در صفحه شطرنج به طوری که هیچ یک به یکدیگر حمله نکنند را توسط الگوریتم کم‌ترین تناقض (Min Conflicts Algorithm) حل کرده‌ایم. همچنین، این مسئله با فرض ۱۶ در ۱۶ بودن ابعاد صفحه و تعداد ۱۶ وزیر نیز حل شده است.

تعریف مسئله: در این مسئله، هر وزیر را در یک ستون جدا در نظر گرفته‌ایم و شماره ردیف هر وزیر را یک متغییر (variable) در نظر می‌گیریم. بنابراین این مسئله دارای هشت متغییر است (برای حالت n=16 ۱۶ متغییر داریم) که هر کدام می‌توانند ۸ مقدار مختلف (برای حالت n=16 ۱۶ مقدار مختلف) اختیار کنند. که مقدار نسبت داده شده به یک متغییر، همان شماره ردیف وزیر مربوط به آن است. حال باید مقادیر این هشت متغییر را جوری انتخاب کنیم که هیچ دو وزیری نتوانند به یکدیگر حمله کنند.

در ادامه به شرح تک تک توابع پیاده‌سازی می‌پردازیم.

- **تابع Conflicts:** این تابع مشخص می‌کند که به ازای یک مقدار (value) مشخص برای یک متغییر مشخص، چه تعداد نقض محدودیت (conflict) خواهیم داشت. اولین ورودی این تابع، متغییر مدنظر است که همان شماره ستون وزیر مربوطه می‌باشد. دومین ورودی آن مقدار مدنظر برای متغییر است که همان شماره ردیف وزیر مربوطه می‌باشد. و آخرین ورودی آن کل صفحه است. این تابع با پیمایش ردیف و قطر مربوط به متغییر ورودی، تعداد تناقضات آن را شمرده و برمی‌گرداند.
- **تابع Initialize**: این تابع صفحه بازی را ایجاد می‌کند و در هر ستون به صورت تصادفی یک وزیر قرار می‌دهد. اگر خانه مقدار صفر داشته باشد یعنی خالی است و اگر مقدار آن یک باشد یعنی یک وزیر در آن قرار دارد.
- **تابع Solution**: این تابع صفحه بازی را دریافت کرده و مشخص می‌کند که آیا حالت صفحه یک حالت هدف است یا خیر. این تابع یک مقدار True یا False بازمی‌گرداند.
- **تابع ConflictedVariables**: این تابع تمام متغییرهایی که شرط محدودیت مسئله نقض می‌کنند پیدا کرده و در یک لیست برمی‌گرداند.
- **تابع MinConflicts**: این تابع الگوریتم Min Conflicts را پیاده‌سازی می‌کند و قسمت اصلی کد است. بدنه آن دقیقا مشابه شبه کد موجود در کتاب راسل پیاده‌سازی شده است. با مطالعه شبه کد الگوریتم در شکل ۱ و نگاه کردن به کد می‌توان به راحتی کارکرد قسمت‌های مختلف کد را متوجه شد.

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
  inputs: csp, a constraint satisfaction problem
          max_steps, the number of steps allowed before giving up

  current ← an initial complete assignment for csp
  for i = 1 to max_steps do
      if current is a solution for csp then return current
      var ← a randomly chosen conflicted variable from csp.VARIABLES
      value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
      set var = value in current
  return failure
```

شکل ۱: شبه کد الگوریتم Min Conflicts

هر چقدر تعداد گام‌های الگوریتم را افزایش دهیم، به دلیل وجود عناصر تصادفی زیاد در الگوریتم، ممکن است جوابی برای مسئله یافت نشود. از طرفی اجرای الگوریتم با گام‌های زیاد، از نظر مدت زمان پردازش به صرفه نیست. بنابراین در بخش main، الگوریتم را آنقدر اجرا می‌کنیم تا به جواب برسیم. طبق تجربه این الگوریتم پس چندین بار اجرا، حتما جواب را خواهد یافت.

لازم به ذکر است که ابعاد صفحه با متغییر n که در اول کد نوشته شده مشخص می‌شود و با تغییر آن، ابعاد صفحه و تعداد وزیران تغییر خواهند کرد. بنابراین اگر جواب مسئله برای هشت وزیر مدنظر است، مقدار n را برابر با هشت قرار داده و اگر مسئله با ۱۶ وزیر مدنظر است، مقدار n را برابر ۱۶ قرار دهید.