توضیحات پروژه csp

آریا ابراهیمی ۹۸۲۲۷۶۲۱۷۵
سارا قوام پور ۹۸۱۲۷۶۲۷۸۱

مدل سازی

- متغیر ها را خانه‌های جدول هستند بنابراین برای یک صفحه $B_{n\times m}$ ، $n\times m$ متغیر خواهیم داشت. از آنجایی که هر کدام از متغیر ها یک قطب آهنربا هستند، یک تابع به نام get_other_pole در نظر گرفته شده که با دریافت مختصات یک قطب، مختصات قطب دیگرش را برمیگرداند و هنگام مقدار دهی این دو قطب را با یکدیگر مقدار میدهیم و در دامنه هایشان هم با یکدیگر آپدیت میشود. میتوانستیم پیاده‌سازی را کلاً به این صورت در نظر بگیریم که هر دو قطب یک متغیر شوند که در این صورت $\frac{n\times m}{2}$ متغیر حاصل میشد.
- دامنه مقادیری است که هر متغیر میتواند اختیار کند که در این مسئله، مقادیر مثبت، منفی و بی مقدار هستند که در کد به صورت '+', '-', ' ' نمایش داده شده اند. همانطور که در قسمت قبل هم توضیح داده شد، اگر متغیری مقدار + بگیرد با استفاده از تابع get_other_pole مختصات قطب دیگرش گرفته می‌شود و مقدار – برای آن اختیار میشود.

توضیحات کد

در قسمت main ورودی ها از فایل دریافت می‌شوند و تابع backtrack صدا زده میشود.
در فایل csp.py توابع مربوطه پیاده‌سازی شده‌اند که به توضیح هر کدام میپردازیم.

- Backtrack : الگوریتم اصلی است که همانند شبه کد زیر عمل میکند:

```
CSP-BACKTRACKING(A)
1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
    a. Add (X←v) to A
    b. If A is valid then
        i. result ← CSP-BACKTRACKING(A)
        ii. If result ≠ failure then return result
    c. Remove (X←v) from A
5. Return failure
```

شکل ۱ : شبه کد backtrack

که برای انتخاب متغیر (X در شبه کد) از الگوریتم MRV استفاده میکنیم و برای ترتیب انتخاب دامنه (D در شبه کد) از الگوریتم LCV استفاده میکنیم.

MRV : روی متغیر ها پیمایش میکند و متغیری که هنوز مقدار دهی نشده است و کمترین دامنه باقی‌مانده را دارد برمیگرداند.
LCV : روی دامنه متغیر انتخاب شده پیمایش میکند و دامنه را بر اساس محدودیت‌هایی که بین همسایه ها و متغیر فعلی و قطب دیگرش ایجاد میشود، به صورت صعودی سورت میکند.

- is_safe : محدودیت‌ها را بررسی میکند، در این مسأله دو سری محدودیت داریم
  ۱. محدودیت‌های دوتایی بین متغیر ها (قطب های همنام نمیتوانند کنار هم باشند)
  ۲. محدودیت‌های سراسری که تعداد بیشتری از متغیر ها را درگیر میکند (برای مثال یک سطر بیشتر از ۲ قطب مثبت نداشته باشد.)

الگوریتم ارائه شده در is_safe، این محدودیت هارا بررسی میکند در ابتدا چک میکند که دو متغیر کنار هم مقدار یکسانی نداشته باشند و اگر چنین بود مقدار False برمیگرداند. در ادامه چک میکند که اگر تمامی متغیرهای یک سطر یا ستون مقدار دهی شده بودند ولی محدودیت مربوط به آن ارضاء نشده بود همانجا False برمیگرداند و در غیر این صورت (متغیرهای سطر یا ستون کامل مقداردهی نشده بودند) چک میکند اگر تعداد مقادیر داده شده برای متغیر ها، محدودیت هارا رد کرد بازهم False برمیگرداند و در آخر اگر هیچکدام از این اتفاق‌ها نیافتاده بود، True برمیگرداند.

- Forward Check: الگوریتم forward check روی همسایه های متغیر مقدار دهی شده پیمایش میکند و مقادیری در دامنه آن‌ها که باعث ایجاد محدودیت می‌شود را حذف میکند. همچنین چک میکند که اگر دامنه یکی از همسایه ها تهی شد، مقدار False ریترن میکند زیرا آن متغیر دیگر نمیتواند مقداری اختیار کند. البته در این مسأله متغیر ها میتوانند مقدار پوچ را در هر مرحله اختیار کنند بنابراین این اتفاق که دامنه متغیر تهی شود اتفاق نمی افتد.

- Ac3: الگوریتم ac3 همانند شبه کد زیر کار میکند و ورژن بزرگتری از forward check میباشد و در محدوده بزرگتری دامنه ها را تغییر میدهد. به دلیل زمان بر بودن این الگوریتم فقط چندبار صدا زده می‌شود (یکبار در ابتدای اجرا و یکبار در وسط اجرای برنامه).

```
AC3
1. contradiction ← false
2. Initialize queue Q with all variables (not yet instantiated)
3. While Q ≠ ∅ and ¬contradiction do
   a. X ← Remove(Q)
   b. For every (not yet instantiated) variable Y related to X
      by a (binary) constraint do
      - If REMOVE-VALUES(X,Y) then
        i. If Y's domain = ∅ then contradiction ← true
        ii. Insert(Y,Q)
```

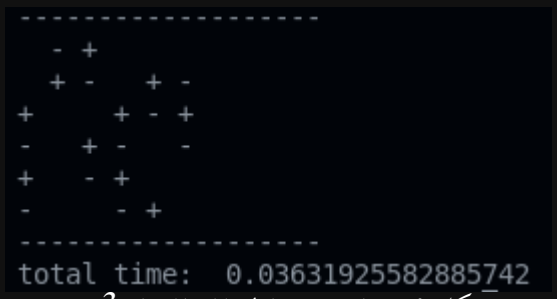
شکل ۲ : الگوریتم *ac3*

```
REMOVE-VALUES(X,Y)
1. removed ← false
2. For every value v in the domain of Y do
   - If there is no value u in the domain of X such
     that the constraint on (X,Y) is satisfied then
     a. Remove v from Y's domain
     b. removed ← true
3. Return removed
```

شکل ۳ : الگوریتم *revise*

شکل ۴ : جواب با استفاده از ac3

شکل ۵: جواب بدون ac3



: شکل ۶: جواب تست اول با استفاده از ac3

با استفاده از الگوریتم ac3 دامنه متغیر ها را کاهش میدهیم و در نتیجه حالات کلی که نیاز به برسی است تا به جواب برسیم کمتر می‌شود و همچنین در زمان پاسخ، تأثیر چندانی ندارد.