« به نام او »

آموزش شبیه سازی دو بعدی فوتبال

نویسندگان : محمد علی میرزایی - علی یعقوبی آزاد

مرجع روبوکاپ ایران

www.iranrcss.com

جلسه سیزدهم

مباحث این جلسه :

۱- جلسه آخر آموزش UVA_Trilearn Base

۲- چگونه مثل یک فرد روبوکاپ کار فکر کنیم ؟

۳- آموزش هوش مصنوعی – روش های ارضای محدودیت

امیدواریم تا به اینجا آموزش ها مفید بوده باشد :

```
١ - SoccerCommand BasicPlayer::directTowards( VecPosition posTurnTo,
    AngDeg angWhenToTurn, VecPosition *pos, VecPosition *vel, AngDeg *angBody  )
٢ - {
٣ -   //  return turnBodyToPoint( posTurnTo );
٤ -   // copy values or initialize when not set
٥ -   VecPosition posAgent= (pos    ==NULL)?WM->getAgentGlobalPosition ():*pos;
٦ -   VecPosition velAgent= (vel    ==NULL)?WM->getAgentGlobalVelocity ():*vel;
٧ -   AngDeg  angBodyAgent= (angBody==NULL)?WM->getAgentGlobalBodyAngle():*angBody;
٨ -   // first predict what happens when the agents rolls out.
٩ -   VecPosition posPred    = WM->predictFinalAgentPos();
١٠ -  AngDeg      angTo      = ( posTurnTo - posPred ).getDirection();
١١ -  AngDeg      ang        = VecPosition::normalizeAngle( angTo - angBodyAgent );
١٢ -  AngDeg      angNeck = ٠;
١٣ -  int iTurn = ٠;
```

```
١٤ - while( fabs( ang ) > angWhenToTurn && iTurn < ٥ )
١٥ - {
١٦ -   iTurn++;
١٧ -   WM->predictStateAfterTurn(
١٨ -        WM->getAngleForTurn( ang, velAgent.getMagnitude() ),
        &posAgent,
        &velAgent,
        &angBodyAgent,
        &angNeck );
١٩ -   ang = VecPosition::normalizeAngle( angTo - angBodyAgent );
٢٠ - }
٢١ - Log.log( ٥٠٩, "direct towards: %d turns", iTurn );
٢٢ - posAgent = (pos    ==NULL)?WM->getAgentGlobalPosition ():*pos;
٢٣ - velAgent = (vel    ==NULL)?WM->getAgentGlobalVelocity ():*vel;
٢٤ - angBodyAgent = (angBody==NULL)?WM->getAgentGlobalBodyAngle():*angBody;
٢٥ - switch( iTurn )
٢٦ - {
٢٧ -   case ٠:   cerr << "direct towards: ٠ turns" ;
```

```
۲۸ -            return SoccerCommand( CMD_ILLEGAL );

۲۹ -    case ۱:

۳۰ -    case ۲:   return turnBodyToPoint( posTurnTo, ۲ );

۳۱ -   default:  return dashToPoint(
          (pos==NULL)?WM->getAgentGlobalPosition ():*pos   );  // stop

۳۲ -  }

۳۳ - }
```

در خط زیر شما تابعی را می بینید که تا به امروز مشاهده نکرده بودید :

```
۹ -   VecPosition posPred    = WM->predictFinalAgentPos();
```

این تابع که جزو توابع predict بیس می باشد ، پیش بینی می کند بازیکن صاحب توپ مکان (Position) بعدیش کجاست . ما توصیه می کنیم اصلا از این توابع استفاده نکنید و خودتون از ابتدا تابعی برای این کار بنویسید . این توابع مشکلات زیادی دارند و این مشکلات باعث عدم پاسخگویی درست به ما می شود .

خواننده عزیز : در این قسمت، سری آموزش های بیس یو وی ای به پایان رسید. گرچه کلیه فایل های بیس بصورت کامل مورد بررسی قرار نگرفت، ولی با اطلاعاتی که در اختیار شما قرار دادیم، توانستیم شما را برای پایه گذاری تیمتان از ابتدا آماده کنیم. هم اکنون شما با ساختار بیس آشنایی دارید و میتوانید با استفاده از آموزش هایی که مطالعه کردید و دانش برنامه نویسیتان، یک تیم از ابتدا بنویسید. مطلبی که همیشه مد نظر

نویسندگان این آموزش ها بوده است، لزوم کار موازی و پژوهش دانشجو در این رشته میباشد. در این قسمت نیز، ما پایان آموزش بیس مقدماتی یو وی ای را اعلام میکنیم و ادامه تحقیقات و کاوش ها را به شما واگذار میکنیم. البته این حرف بدین معنی نیست که دیگر مبحثی برای فراگیری در شبیه سازی فوتبال دوبعدی وجود ندارد! تیم مرجع روبوکاپ ایران، قدرتمند مثل همیشه و بعنوان پیشرو در ارائه آموزش های جامع و عملی روبوکاپ، تهیه مقالات مختلف و آموزش های گوناگون در زمینه آموزش سایر بیس ها، مباحث پیشرفته هوش مصنوعی و برنامه نویسی، ارائه مقالات کاربردی در کلیه رشته های روبوکاپ و ... را در برنامه خود گنجانده است.

شما در هر زمان، به هر مشکلی برخورد کردید، میتوانید آنرا در انجمن تخصصی روبوکاپ ایران (http://www.forum.iranrcss.com) مطرح نمایید. تیم مرجع روبوکاپ ایران، همواره پاسخگوی سوالات شماست.

برای شما آرزوی موفقیت از خدای منان را داریم.

چگونه مثل یک فرد روبوکاپ کار فکر کنیم؟

ما در طی این آموزش هایی که تا به این روز نوشتیم ، سعی داشتیم تنها یک شمای کلی از بیس به شما نشان بدهیم . ما اصلا هدفی مبنی بر آموزش بیس UVA به طور کامل نداشتیم و نداریم . در روبوکاپ، استاد معنی ندارد و باید خودتان تلاش کنید . هیچوقت از یک کسی که روبوکاپ کار می کند ، این انتظار را نداشته باشید که به شما خط به خط چیزی را توضیح بدهد . این آموزش ها هم تنها برای کامل بودن مقالات نوشته شد و گرنه ما نیز بر این عقیده هستیم که اینگونه نمی شود به پیشرفت روبوکاپ کمک کرد . ما نباید بیس را بفهمیم ، ما باید شبیه سازی دوبعدی را بفهمیم . ما نباید فکر کنیم آقای یله کوک و یا آقای آکیاما چه در ذهن داشته اند ، باید فکر کنیم ما برای این موقعیت چه فکری داریم . به طور مثال نباید فکر کنیم چرا آقای آکیاما شوت تیمش را اینگونه نوشته است ، باید فکر کنیم ما چه ایده ای در مورد شوت کردن داریم و در خیلی از موارد دیگر ...

در سال های اخیر ما هیچ بیس جدیدی نداشتیم و همه از بیس های موجود استفاده می کنند و این یعنی شکست ؛ یعنی هرگز نرسیدن به هدف اصلی . در حال حاضر مسابقه جهانی ، مسابقات آزاد و حتی خوارزمی ، دیگر براساس برد و یا باخت تیم ها جایزه نمی دهند بلکه به فکری که پشت تیم است امتیاز می دهند . برای روبوکاپ مهم نیست شما ۲۰ گل به تیم مقابل بزنید ؛ مهم این است که شما چه ابداع جدیدی برای تیمتان انجام داده اید . اینکه من روی بیس کسی کار کنم و به هر تیمی ۲۰ گل بزنم مهم نیست ، مهم این است که من بیس خودم را با ابداعات خودم بنویسم . نتیجه بازی مهم نیست ، مهم ابداع است . مهم فکر است و نه چیز دیگر .

علت این که همواره آموزش های بیس از هوش مصنوعی کمتر بود هم همین است . ما باید هوش تیم را بالا ببریم نه اینکه به بازیکن دستور ماشینی بدیم . اگر شماره ۱۰ بودی به بازیکن شماره ۴ پاس بده(مثال) ... این

برنامه غلط است و ما باید با توجه به موقعیت بازی کنیم . شما هیچگاه در واقعیت قبل از بازی همچین دستوری به بازیکنان تان نمی دهید . اینجا محیط شبیه سازی شده است ، اما تمام کار هایی که در واقعیت میکنید را باید پیاده سازی کنید . گاهی پیش آمده شما در بازی سریع نقشه ی بازی را عوض کنید ؛ در شبیه سازی دو بعدی هم باید همینگونه باشید . نباید تیمتان تنها برای یک موقعیت نوشته شده باشد .

همیشه برای بهترینها بجنگید و آمادگی مواجه با مشکلات را داشته باشید . هیچگاه از شکست ناامید نشوید. من اولین باری که یک مسابقه دادم ۱۲ بر صفر شکست خوردم و همه اش بر اساس یک اشتباه از جانب خودم بود (!!!) . اگر من آنروز می خواستم روبوکاپ را رهایش کنم در حال حاضر این آموزش ها را نمی نوشتم . هیچوقت این جمله ی زیبا که " هر شکستی مقدمه ای برای پیروزی است " را فراموش نکنید . همیشه از شکست هایتان درس بگیرید . همیشه بازی کردن تیمتان را تحلیل کنید . یک تیم خوب ، یک تحلیلگر خوب هم دارد پس به یک تحلیلگر نیاز دارید . باید سایکل به سایکل بازی را بررسی کنید . باید مو را از ماست بیرون بکشید !

پشتکار داشته باشید . روبوکاپ احتیاج به یک روحیه ی قوی و پشتکار بالا دارد . همیشه سعی کنید خسته نشوید . " خستگی و شما ؟!!! " باید نشانتان باشد .

نتیجه نباید شما را مایوس کند . اینکه شما در حال کمک به جامعه ی روبکاپ هستید باید همیشه قوت قلبی برای شما باشد .

امروز آخرین جلسه ای بود که ما به طور اختصاصی به بحث در مورد بیس UVA پرداختیم . از جلسه ی بعد به ادامه ی توضیحات در مورد اینکه ، چگونه مثل یک فرد روبوکاپ کار فکر کنیم ، خواهیم پرداخت . لطفا هرگونه سوالی و یا انتقاد و پیشنهادی داشتید به ما انتقال بدهید. ما با نظرات شما پیشرفت خواهیم کرد .

باتشکر

گروه شبیه سازی دوبعدی مرجع روبوکاپ ایران

مسائل ارضای محدودیت

فهرست :

ارضای محدودیت (CSP) چیست؟

- مجموعه متناهی از متغیرها؛ $X_1, X_2, ..., Xn$
- مجموعه متناهی از محدودیتها؛ $C_1, C_2, ..., Cm$
- دامنه های ناتهی برای هر یک از متغیرها؛ $DX_1,DX_2,...,DXn$
- هر محدودیت Ci زیرمجموعه ای از متغیرها و ترکیبهای ممکنی از مقادیر برای آن زیرمجموعه ها

- هر حالت با **انتساب** مقادیری به چند یا تمام متغیرها تعریف میشود
- انتسابی که هیچ محدودیتی را نقض نکند، انتساب **سازگار** نام دارد
- انتساب **کامل** آن است که هر متغیری در آن باشد
- **راه حل** CSP یک انتساب کامل است اگر تمام محدودیتها را برآورده کند
- بعضی از CSPها به راه حلهایی نیاز دارند که **تابع هدف** را بیشینه کنند

مثال CSP: رنگ آمیزی نقشه

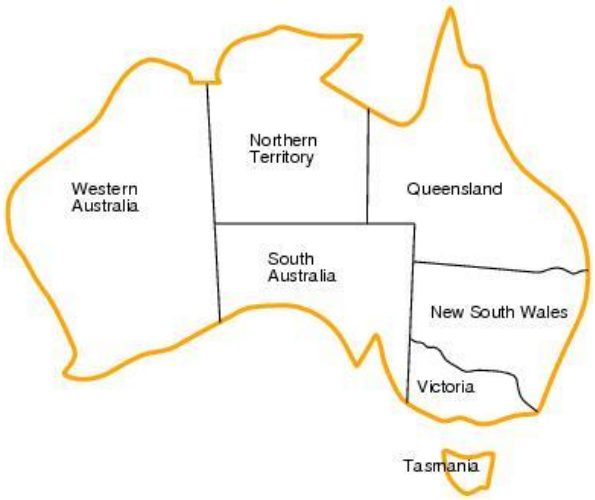
متغیرها: WA, NT, Q, NSW, V, SA, T

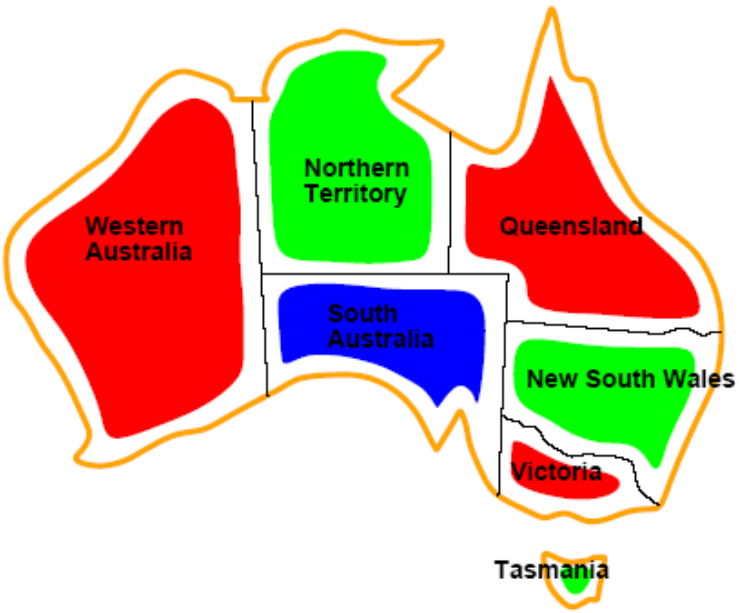
دامنه: Di = {آبی، سبز، قرمز}

محدودیتها: دو منطقه مجاور، همرنگ نیستند

مثال: WA ≠ NT یعنی (WA,NT) عضو

{(قرمز,سبز),(قرمز,آبی),(سبز,قرمز)، (سبز,آبی),(آبی,قرمز),(آبی,سبز)}

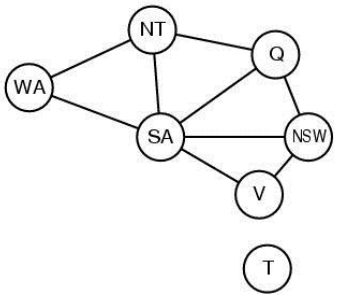




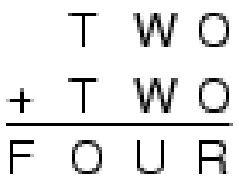
راه حل انتساب مقادیری است که محدودیتها را ارضا کند.

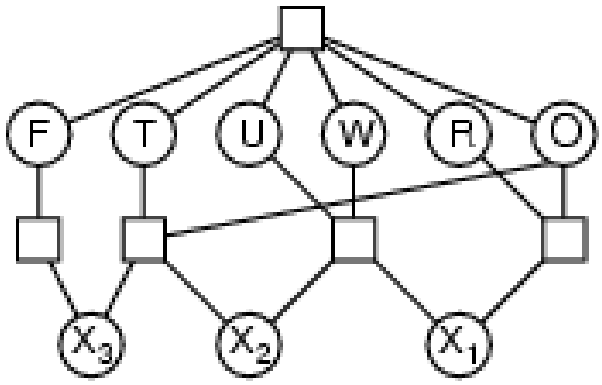
گراف محدودیت:

- **در گراف محدودیت:**
  - **گره ها: متغیرها**
  - **یالها: محدودیتها**
- **گراف برای ساده تر کردن جست و جو بکار میرود.**



مثال CSP: رمزنگاری





متغیرها:F,T,U,W,R,O,X۱,X۲,X۳ دامنه:{۹و۸و۷و۶و۵و۴و۳و۲و۱و۰}

محدودیتها: F,T,U,R,O,W مخالفند - O+O=R+۱۰.X۱ - ...

***

نمایش حالتها در CSP از الگوی استانداردی پیروی میکند.

برای CSP میتوان فرمول بندی افزایشی ارائه کرد:

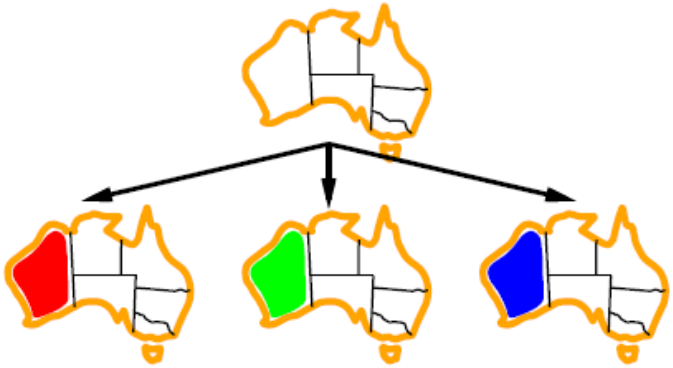
- حالت اولیه: انتساب خالی{} که در آن، هیچ متغیری مقدار ندارد
- تابع جانشین: انتساب یک مقدار به هر متغیر فاقد مقدار، به شرطی که با متغیرهایی که قبلا مقدار گرفتند، متضاد نباشند
- آزمون هدف: انتساب فعلی کامل است
- هزینه مسیر: هزینه ثابت برای هر مرحله

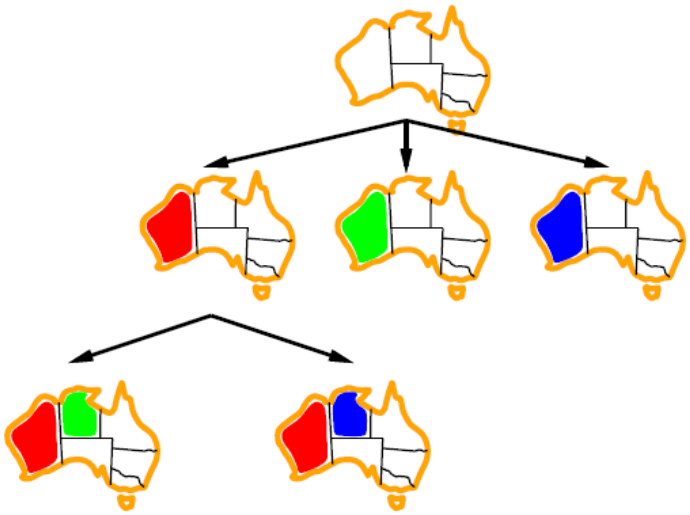
جست و جوی عقبگرد برای CSP

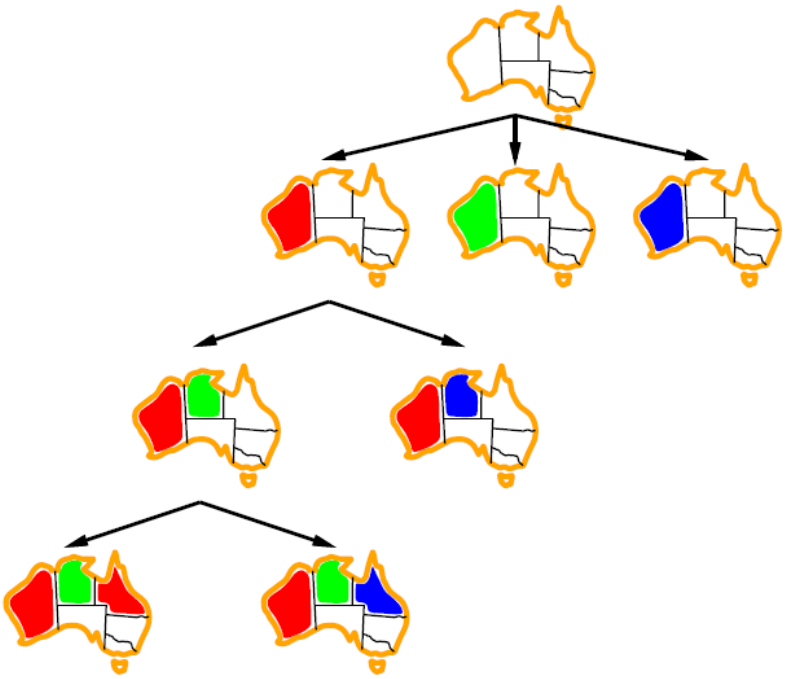
- جست و جوی عمقی
- اتخاب مقادیر یک متغیر در هر زمان و عقبگرد در صورت عدم وجود مقداری معتبر برای انتساب به متغیر
- یک الگوریتم ناآگاهانه است.

➢ برای مسئله های بزرگ کارآمد نیست.

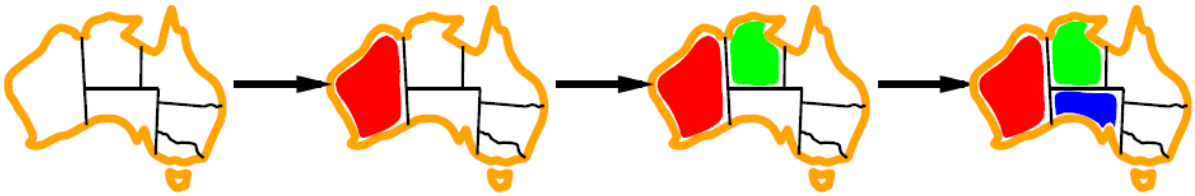
مثال جست و جوی عقبگرد برای CSP





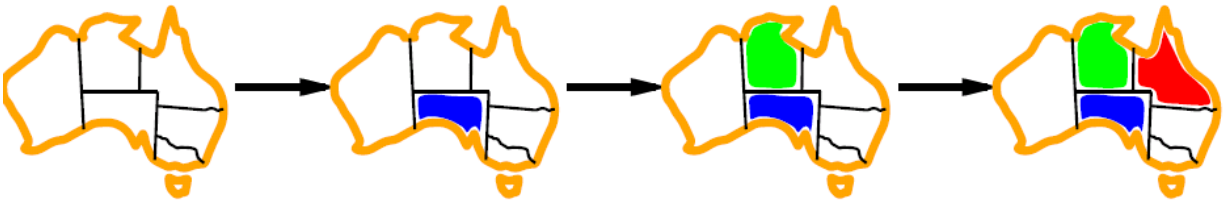


مقادیر باقیمانده کمینه(MRV)



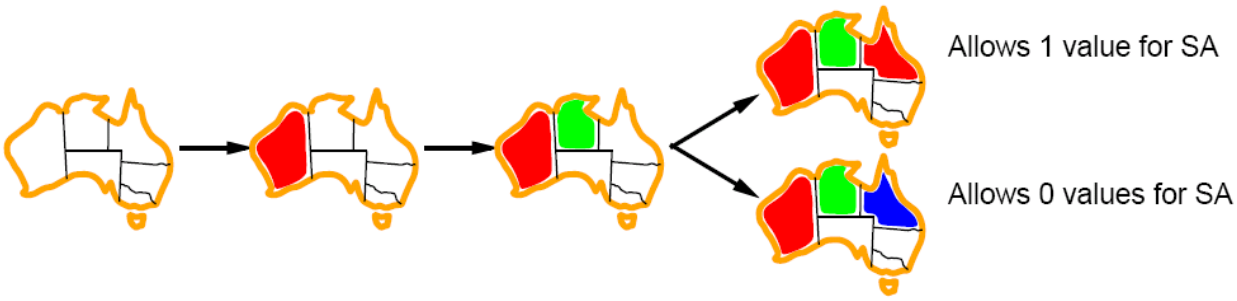
- انتخاب متغیری با کمترین مقادیر معتبر
- متغیری انتخاب میشود که به احتمال زیاد، بزودی با شکست مواجه شده و درخت جست و جو را هرس میکند.

اکتشاف درجه ای



- سعی میکند فاکتور انشعاب را در انتخاب آینده کم کند.
- متغیری انتخاب میکند که در بزرگترین محدودیتهای مربوط به متغیرهای بدون انتساب قرار دارد.

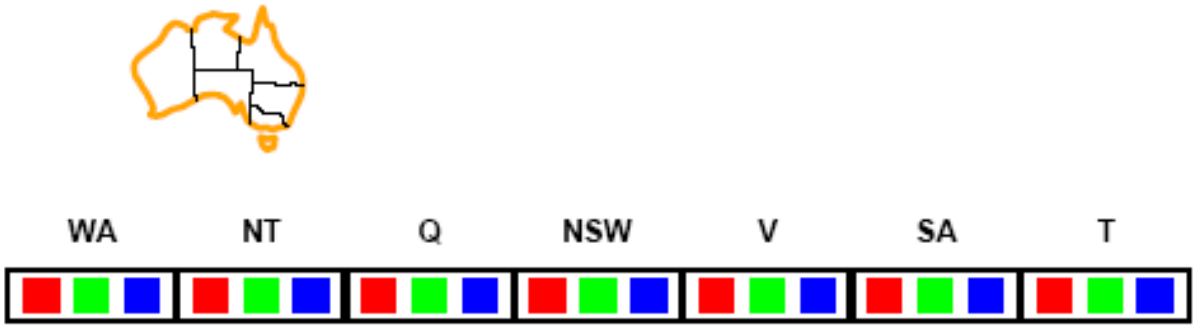
اکتشاف مقداری باکمترین محدودیت

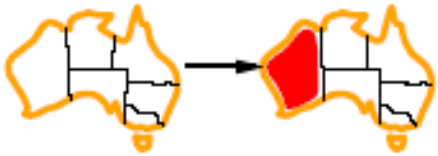


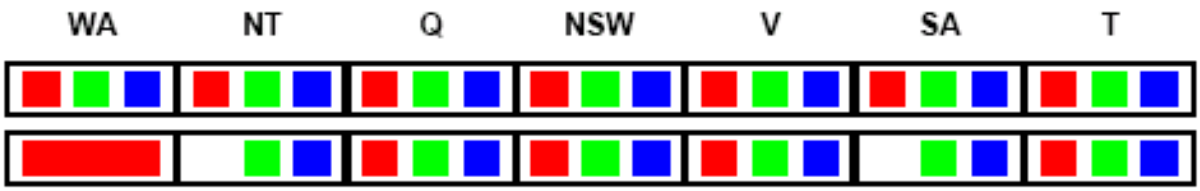
- این روش مقداری را ترجیح میدهد که در گراف محدودیت، متغیرهای همسایه به ندرت آن را انتخاب میکنند.
- سعی بر ایجاد بیشترین قابلیت انعطاف برای انتساب بعدی متغیرها.

بررسی پیشرو

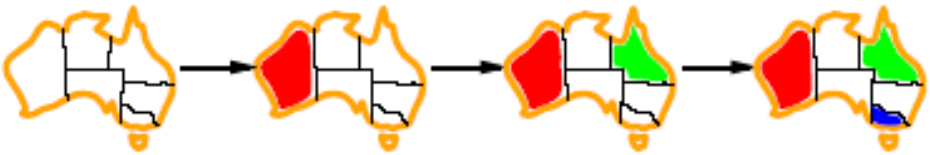
وقتی انتساب به X صورت میگیرد، فرایند بررسی پیشرو، متغیرهای بدون انتساب مثل Y را در نظر میگیرد که از طریق یک محدودیت به X متصل است و هر مقداری را که با مقدار انتخاب شده برای X برابر است، از دامنه Y حذف میکند.





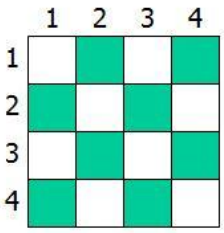

WA
NT
Q
NSW
V
SA
T

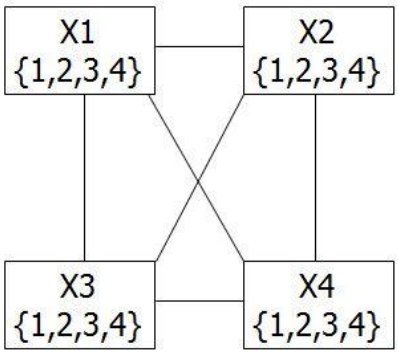

WA
NT
Q
NSW
V
SA
T

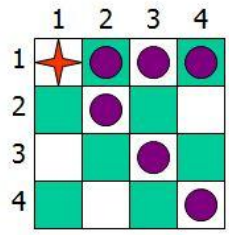


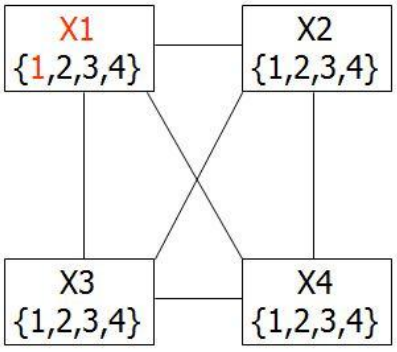

WA
NT
Q
NSW
V
SA
T

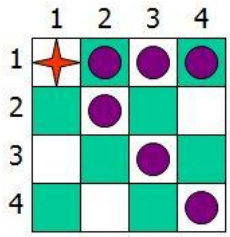
مثال: مسئله ۴-وزیر

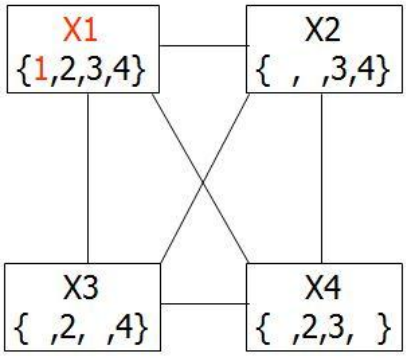


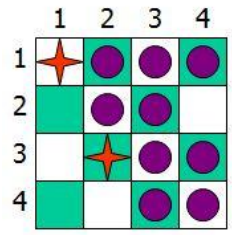


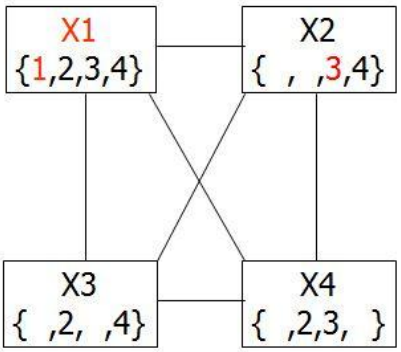


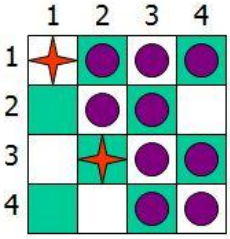


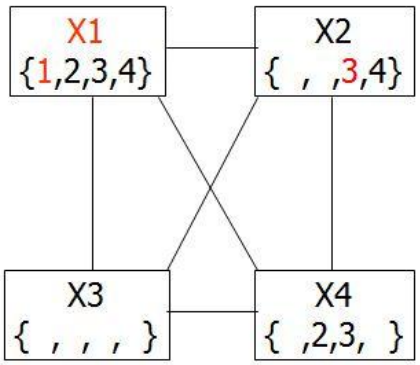


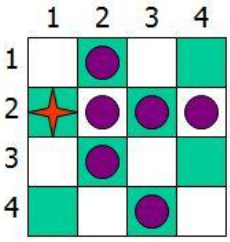


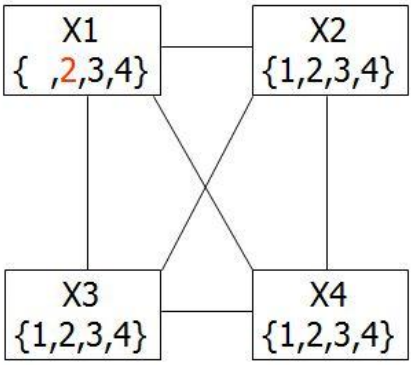

1 2 3 4
1
2
3
4

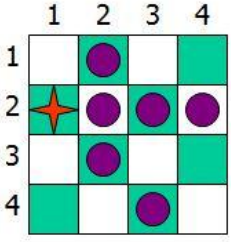

X1
{1,2,3,4}
X2
{ , ,3,4}
X3
{ ,2, ,4}
X4
{ ,2,3, }

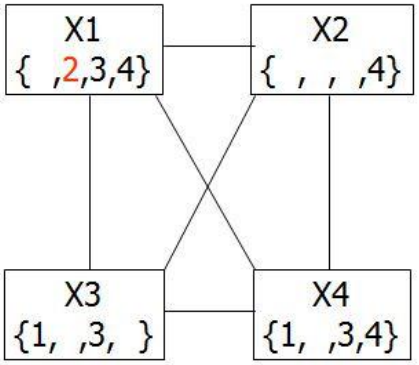

1 2 3 4
1
2
3
4

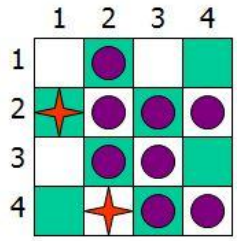

X1
{1,2,3,4}
X2
{ , ,3,4}
X3
{ , , , }
X4
{ ,2,3, }

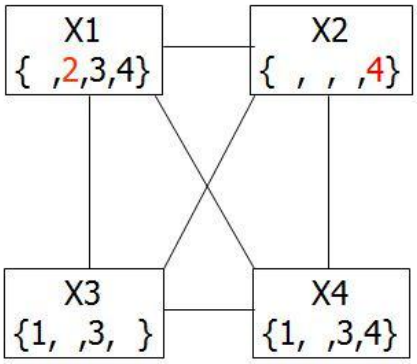

1 2 3 4
1
2
3
4

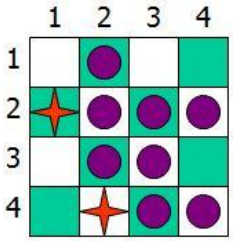

X1
{ ,2,3,4}
X2
{1,2,3,4}
X3
{1,2,3,4}
X4
{1,2,3,4}

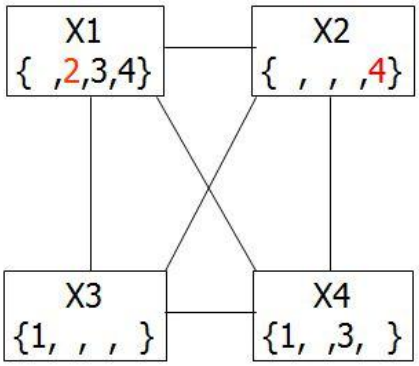

1 2 3 4
1
2
3
4

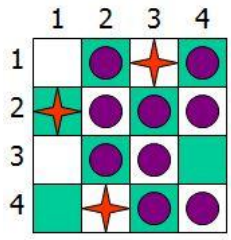

X1
{ ,2,3,4}
X2
{ , , ,4}
X3
{1, ,3, }
X4
{1, ,3,4}

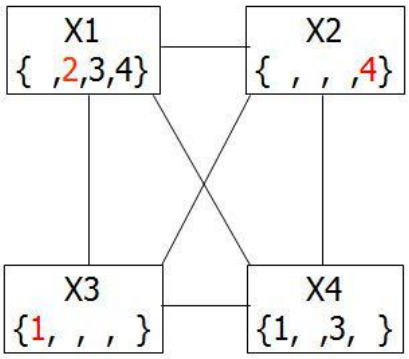

1 2 3 4
1
2
3
4

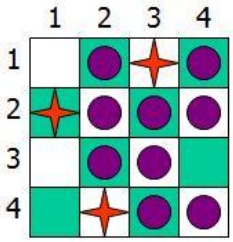

X1
{ ,2,3,4}
X2
{ , , ,4}
X3
{1, ,3, }
X4
{1, ,3,4}

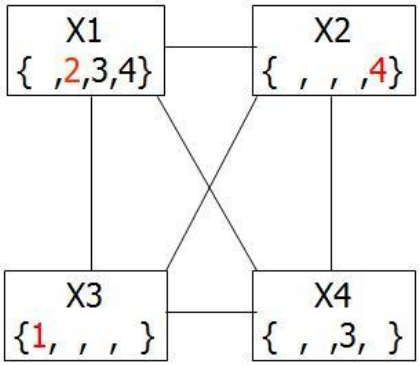

1 2 3 4
1
2
3
4

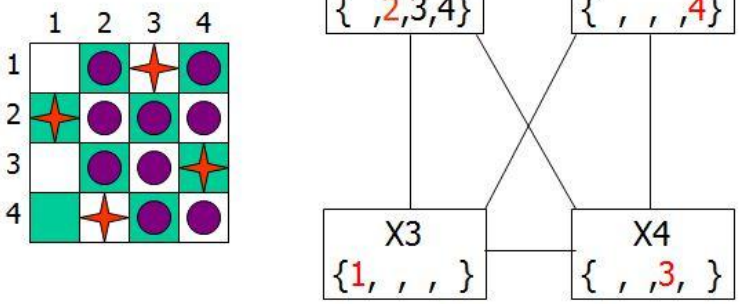

X1
{ ,2,3,4}
X2
{ , , ,4}
X3
{1, , , }
X4
{1, ,3, }


1 2 3 4
1
2
3
4


X1
{ ,2,3,4}
X2
{ , , ,4}
X3
{1, , , }
X4
{1, ,3, }


1 2 3 4
1
2
3
4

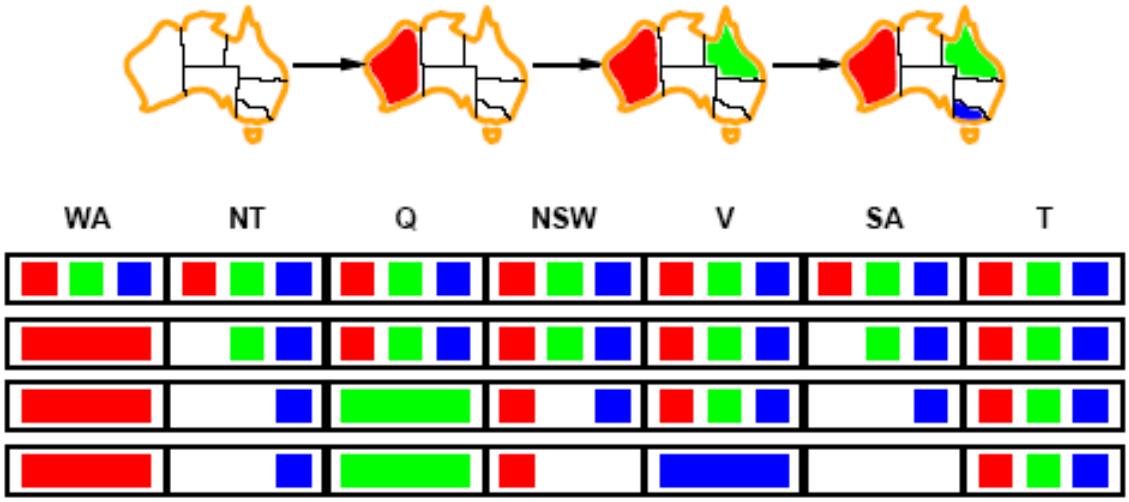

X1
{ ,2,3,4}
X2
{ , , ,4}
X3
{1, , , }
X4
{ , ,3, }



پخش محدودیت

پخش الزام محدودیتهای یک متغیر به متغیرهای دیگر

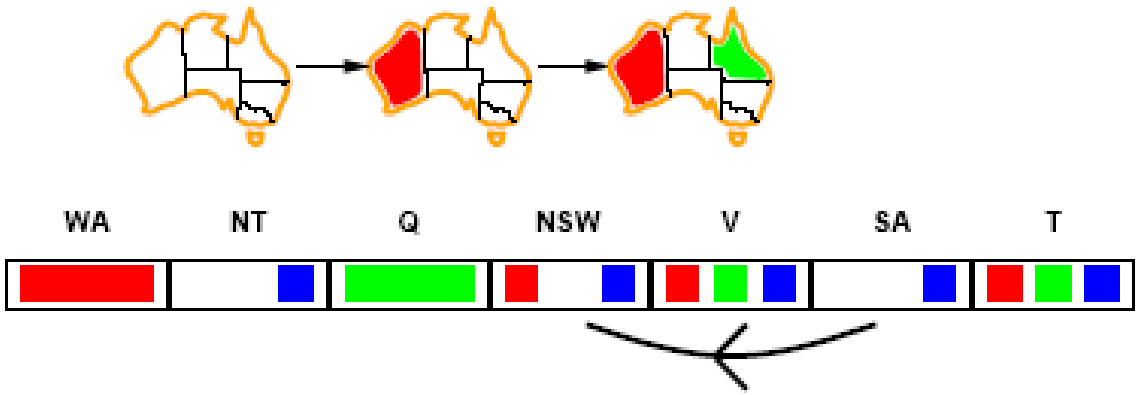
- مثال: پخش محدودیتهای WA و Q به NT و SA



سازگاری یال

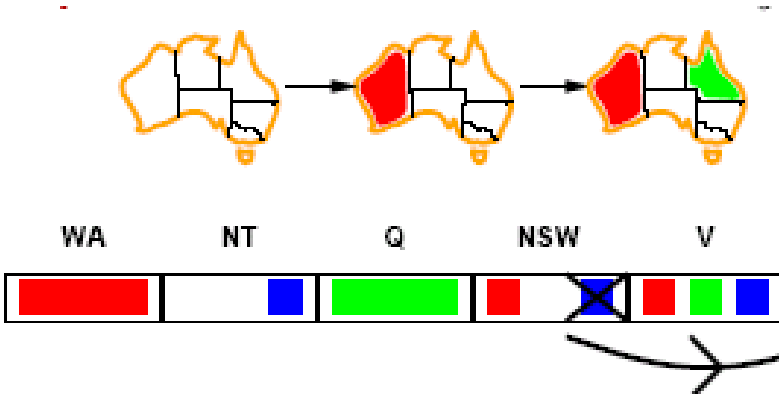
- روش سریعی برای پخش محدود و قویتر از بررسی پیشرو
- یال؛ یال جهت دار در گراف محدودیت
- بررسی سازگاری یال
  - یک مرحله پیش پردازش، قبل از شروع جستجو
  - یک مرحله پخشی پس از هر انتساب در حین جستجو

مثال: سازگاری یال



$SA \rightarrow NSW$ سازگار است اگر

SA=blue and NSW=red

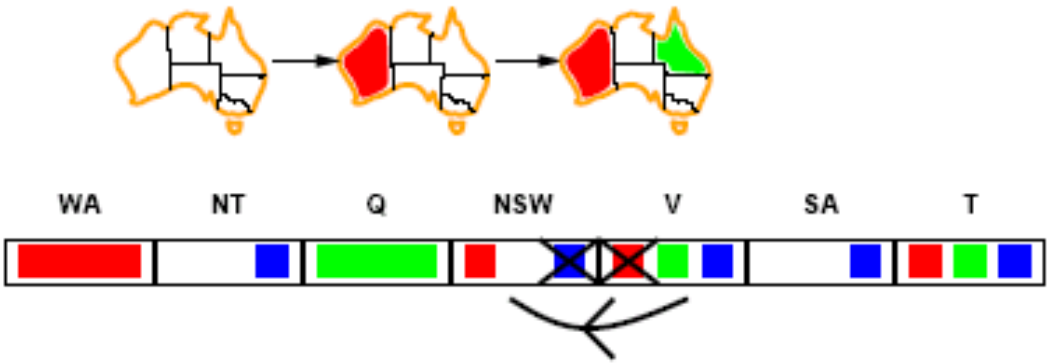


NSW $\rightarrow$ SA سازگار است اگر

SA=blue and NSW=red

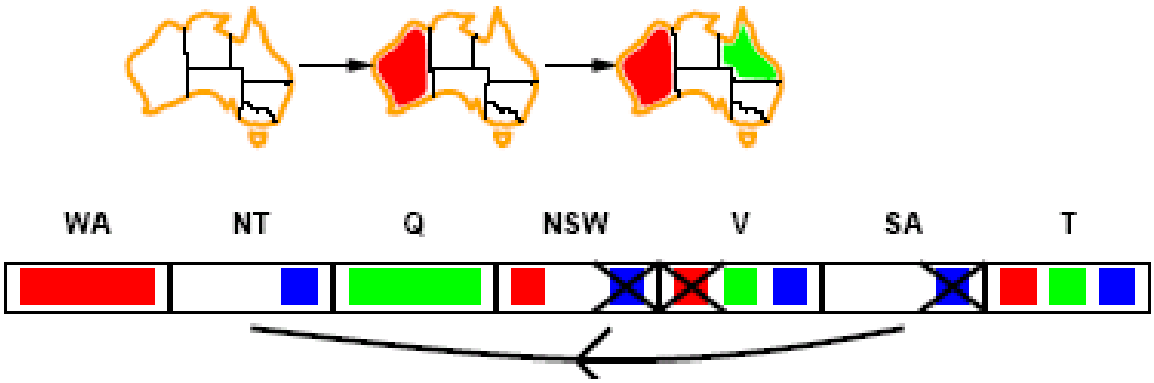
NSW=blue and SA=???

یال میتواند سازگار شود با حذف blue از NSW



یال میتواند سازگار شود با حذف blue از NSW

حذف red از V



یال میتواند سازگار شود با حذف blue از NSW

حذف red از V

تکرار تا هیچ ناسازگاری باقی نماند.

***

سازگاری K

- سازگاری یال تمام ناسازگاریهای ممکن را مشخص نمیکند.
- با روش سازگاریK، شکلهای قویتری از پخش را میتوان تعریف کرد.
- در صورتی CSP سازگاریK است، که برای هر k-۱ متغیر و برای هر انتساب سازگار با آن متغیرها، یک مقدار سازگار، همیشه بتواند به متغیر kام نسبت داده شود.
- بطور مثال:
  - سازگاری۱: هر متغیر با خودش سازگار است(سازگاری گره)
  - سازگاری۲: مشابه سازگاری یال

- سازگاریk: بسط هر جفت از متغیرهای همجوار به سومین متغیر همسایه(سازگاری مسیر)

- گراف در صورتی قویا سازگارK است که:
  - سازگارk باشد.
  - همچنین سازگار$k-1$ و سازگار$k-2$ و... سازگار ۱ باشد.
- در این صورت، مسئله را بدون عقبگرد میتوان حل کرد.
- پیچیدگی زمانی آن $O(nd)$ است.

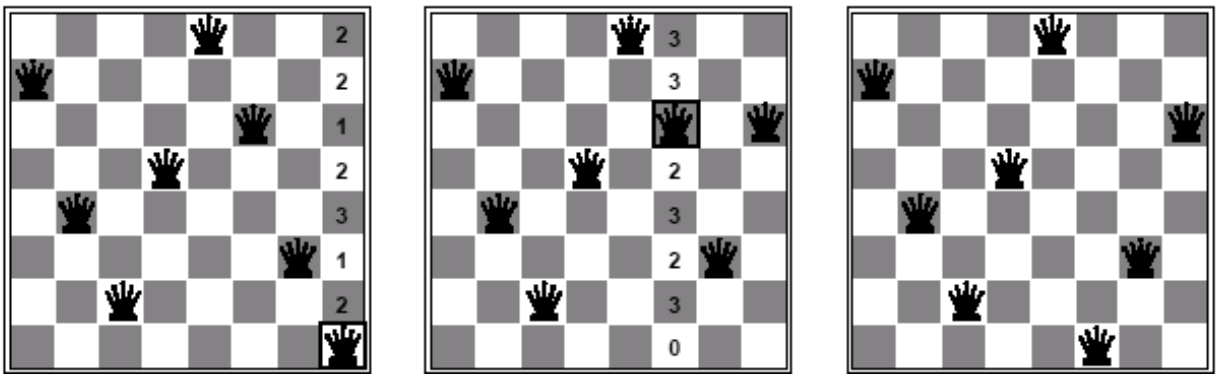
جست و جوی محلی در مسائل ارضای محدودیت

- **بسیاری از CSPها را بطور کارآمد حل میکنند.**
  - **حالت اولیه، مقداری را به هر متغیر نسبت میدهد.**
  - **تابع جانشین، تغییر مقدار یک متغیر در هر زمان**
- **انتخاب مقدار جدید برای یک متغیر**
  - **انتخاب مقداری که کمترین برخورد را با متغیرهای دیگر ایجاد کند(اکتشاف برخورد کم).**
  - **زمان اجرای برخورد کم مستقل از اندازه مسئله است.**
  - **برخورد کم، برای مسئله های سخت نیز کار میکند.**

- جست و جوی محلی میتواند در صورت تغییر مسئله، تنظیمات **Online** را انجام دهد.

❖ مثال:

راه حل دو مرحله ای برای مسئله ۸وزیر با استفاده از کمترین برخورد



- در هر مرحله، یک وزیر برای انتساب مجدد در ستون خودش انتخاب میگردد.
- تعداد برخوردها در هر مربع نشان داده شده است.
- الگوریتم وزیر را به مربعی با کمترین برخورد انتقال میدهد، بطوریکه گره ها را بطور تصادفی میشکند.

در قسمت بعد، به مبحث **جستجوی خصمانه** میپردازیم.

پایان قسمت سیزدهم