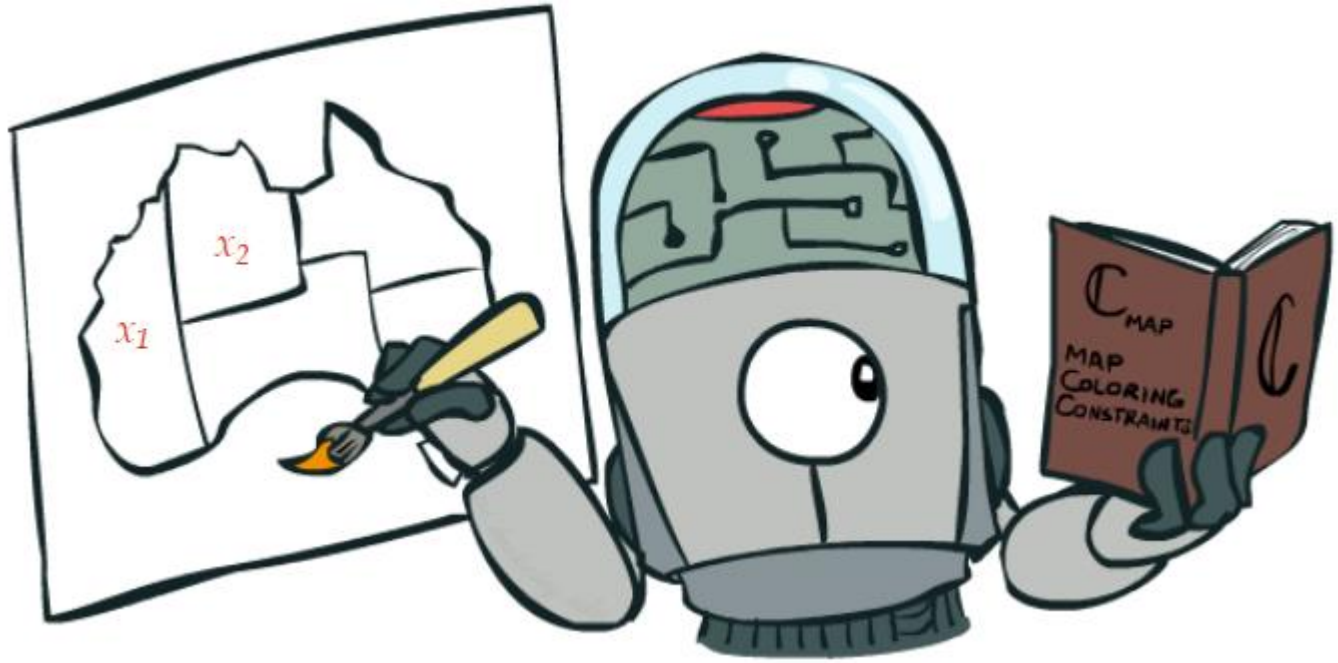
هوش مصنوعی
(مسائل ارضای محدودیت)

صادق اسکندری – دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir



مقدمه

مسائل جستجوی برنامه ریزی (Planning)

- حالت اولیه و حالت هدف مشخص هستند.
- چیزی که اهمیت دارد: مسیر از حالت اولیه به هدف
- مسیرها دارای هزینه ها و عمق های مختلفی هستند.
- مثال: مسیریابی، 8-puzzle

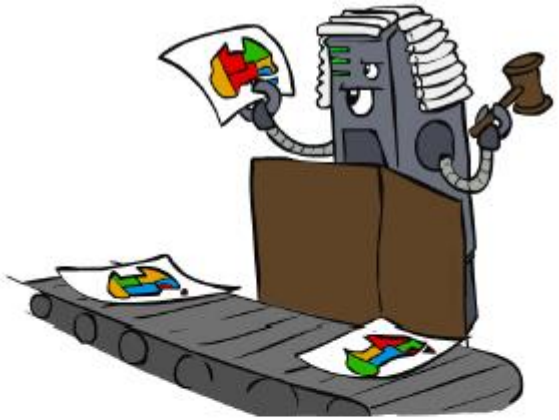
مسائل جستجوی شناسایی (Identification)

- خود هدف برای ما اهمیت دارد، نه مسیر رسیدن به هدف
- مسائل ارضای محدودیت (Constraint Satisfaction Problems) نوع خاصی از مسائل شناسایی هستند.
- مثال: مسائل رنگ آمیزی گراف، هشت وزیر، فروشنده دوره گرد و ...

مسائل ارضای محدودیت (CSP)

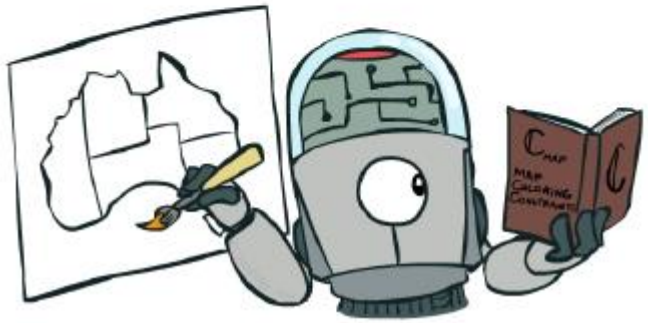
مسائل جستجوی استاندارد

- یک حالت می تواند هر ساختاری داشته باشد.
- تابع تست هدف، می تواند هر تابعی روی حالت ها باشد.
- تابع بعدی (Successor) می تواند هر تابعی باشد.



مسائل CSP

- یک حالت خاص از مسائل جستجو است.
- حالت ها به وسیله متغیرهای X_i با مقادیری از دامنه D تعریف می شوند.
- تست هدف، مجموعه ای از قیود (Constraints) است که ترکیبات قابل قبول از مقادیر برای زیرمجموعه هایی از متغیرها را مشخص می کنند.



مثالهایی از CSP: رنگ آمیزی نقشه

متغیرها: $WA,\ Q,\ NT,\ WA,\ T,\ SA,\ V$

دامنه ها: $D = \{red, green, blue\}$

قیود: نواحی مجاور باید دارای رنگ های مختلفی باشند.

شیوه بیان قیود

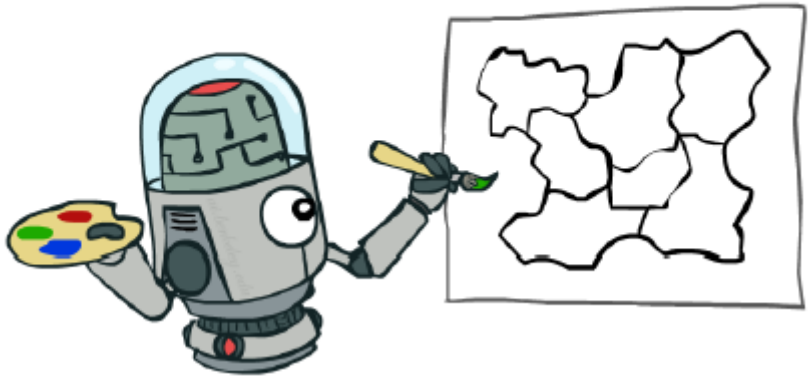
ضمنی: $NT \neq SA, WA \neq SA, WA \neq NT, \ldots$

صریح:

$$(WA, NT) \in \{(r,g),(r,b),(g,r),(g,b),(b,r),(b,g)\}$$

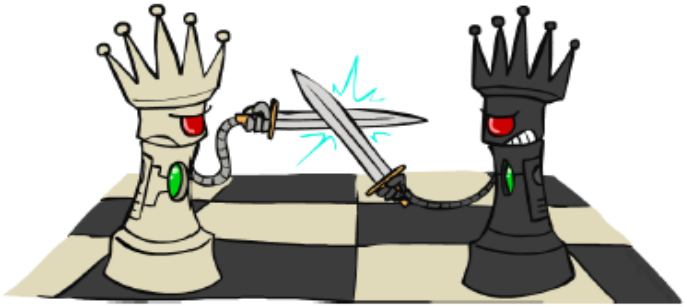
$$(WA, T) \in \{(r,r),(b,b),(g,g),(g,b),(b,r),(b,g),(r,g),(r,b),(g,r),(g,b),(b,r),(b,g)\}$$

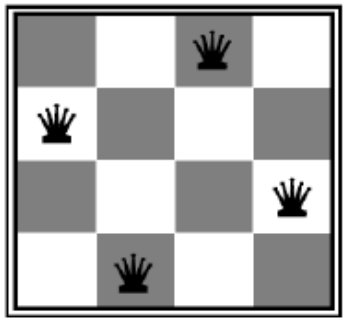
...



یک پاسخ (Solution) برای این مسئله می تواند به صورت {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green} باشد.

مثالهایی از CSP: N-وزیر





فرموله سازی شماره ۱

متغیرها: خانه واقع در سطر i ام و ستون j ام X_{ij}

دامنه ها: $\{0,1\}$

$X_{ij} = 0$ در خانه واقع در سطر i ام و ستون j ام وزیر قرار ندارد.

$X_{ij} = 1$ در خانه واقع در سطر i ام و ستون j ام وزیر قرار دارد.

قیود

۱- تعداد کل وزیران برابر N است. $\sum_{i,j} X_{ij} = N$

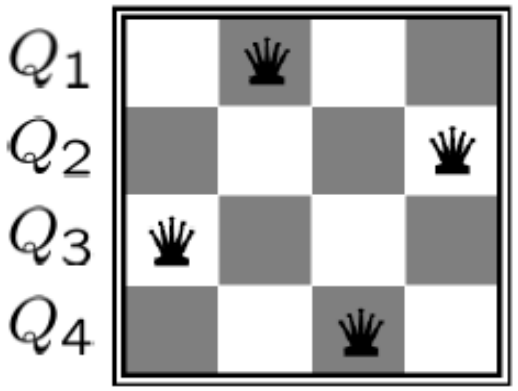
۲- در هر سطر حداکثر یک وزیر قرار می گیرد. $\forall i,j,k \ (X_{ij}, X_{ik}) \in \{(0,0),(0,1),(1,0)\}$

۳- در هر ستون حداکثر یک وزیر قرار می گیرد. $\forall i,j,k \ (X_{ij}, X_{kj}) \in \{(0,0),(0,1),(1,0)\}$

٤- در هر قطر (اصلی و فرعی) حداکثر یک وزیر قرار می گیرد. $\forall i,j,k \ (X_{ij}, X_{(i+k)(j+k)}) \in \{(0,0),(0,1),(1,0)\}$

$\forall i,j,k \ (X_{ij}, X_{(i+k)(j-k)}) \in \{(0,0),(0,1),(1,0)\}$

مثالهایی از CSP: N-وزیر



فرموله سازی شماره ۲

متغیرها: سطر k ام Q_k

دامنه ها: $\{1,2,3, \dots ,N\}$ (موقعیت وزیر در سطر k ام)

قیود

ضمنی: به ازای هر i و j؛ Q_i و Q_j یکدیگر را تهدید نکنند.

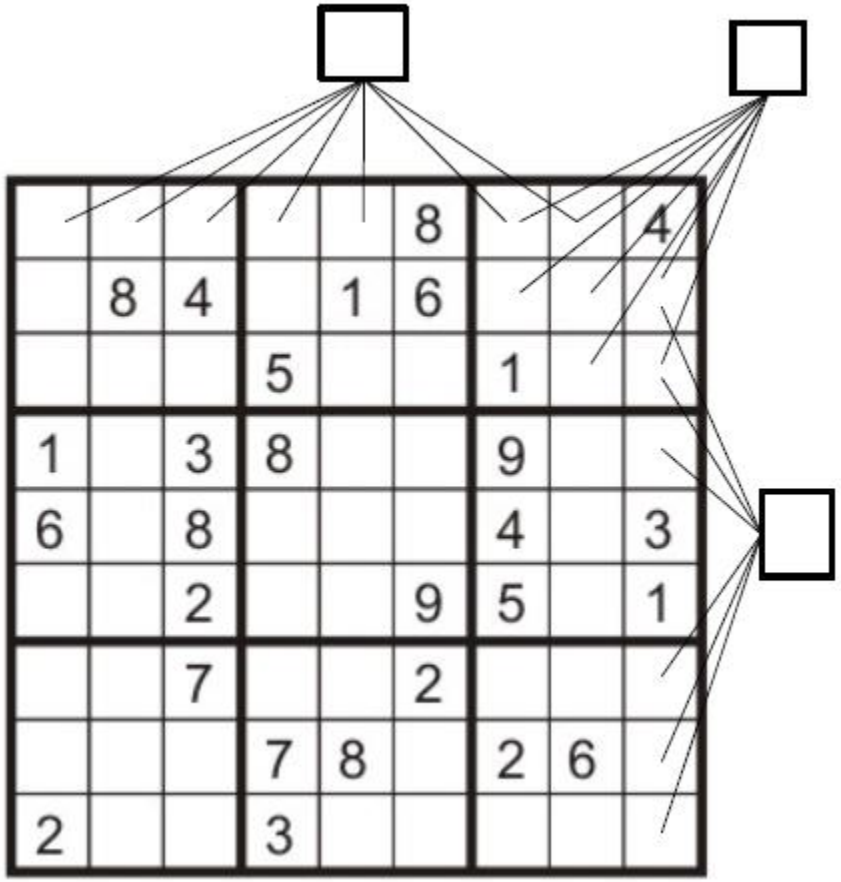
صریح:

$(Q_1, Q_2) \in \{(1,3), (1,4), \dots\}$

$(Q_1, Q_3) \in \{(1,2), (1,4), \dots\}$

$\vdots$

مثالهایی از CSP: Sudoku



متغیرها: هر مربع خالی

دامنه ها: $\{1, 2, 3, \dots, 9\}$

قیود

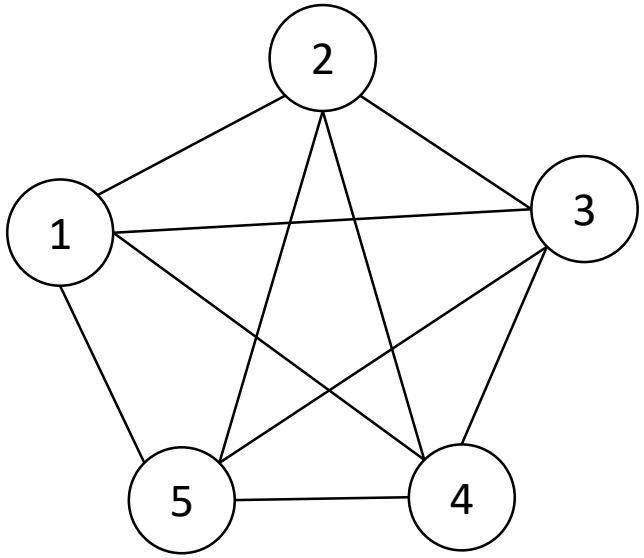
۱- تمامی ۹ مربع در هر ستون دارای مقادیر متفاوتی باشند.

۲- تمامی ۹ مربع در هر سطر دارای مقادیر متفاوتی باشند.

۳- تمامی ۹ مربع در هر ناحیه دارای مقادیر متفاوتی باشند.

مثالهایی از CSP: فروشنده دوره گرد (TSP)

یک فروشنده، باید از N شهر عبور کرده و به نقطه شروع برگردد. از هر شهر یک بار و فقط یک بار عبور کند.



متغیرها: $x_1, x_2, \dots, x_N$

x_i: شماره i امین شهر در مسیر

یا

x_i: اولویت شهر i ام

دامنه ها: $\{1, 2, 3, \dots, N\}$

قیود:

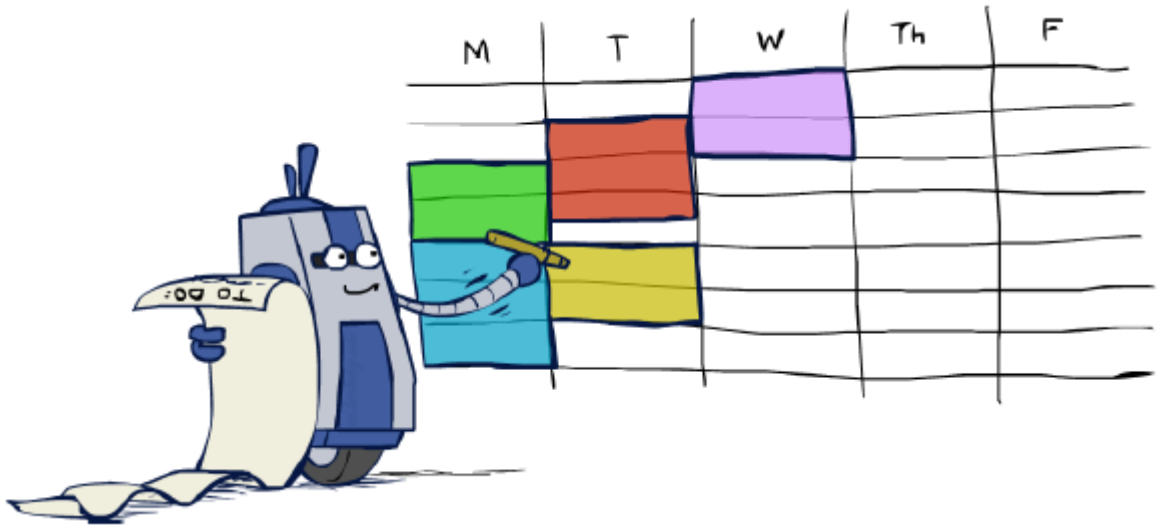
$\forall i, j: x_i \neq x_j$

$\sum_{k=1}^{N} w(x_k, x_{k+1}) + w(x_n, x_1)$ کمینه باشد.

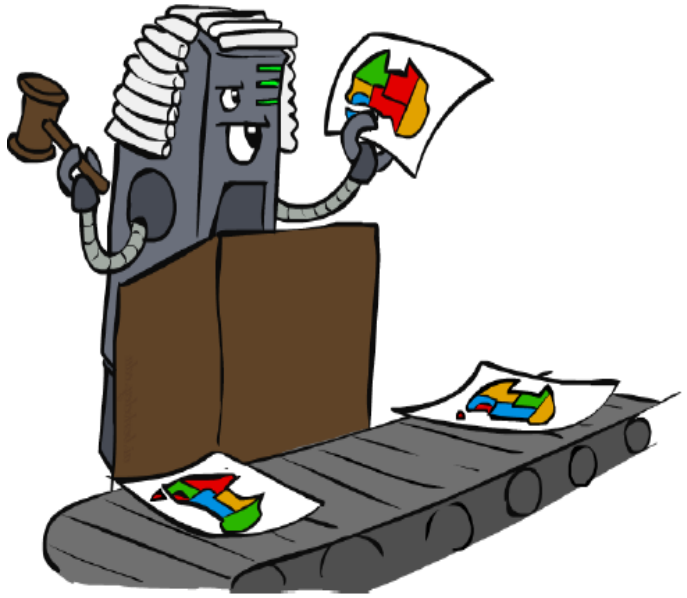
مثالهایی از CSP

مثالهای دیگر:

- مسائل تخصیص: چه کسی چه درسی را تدریس کند.
- جداول زمانبندی: چه درسی در چه کلاسی و چه ساعتی ارائه شود.
- زمانبندی اتوبوس ها
- ⋮



فرموله سازی جستجو در مسائل CSP



حالت ها بر اساس مقادیری که به متغیرها تخصیص داده شده اند، تعریف می شوند.

- یک حالت برای ٤ وزیر: $\{Q_1 = 1, Q_2 = 4, Q_3 = 2, Q_4 = 3\}$
- یک دیگر حالت برای ٤ وزیر: $\{Q_1 = 1, Q_2 = null, Q_3 = null, Q_4 = null\}$

حالت اولیه همان تخصیص تهی {} است.

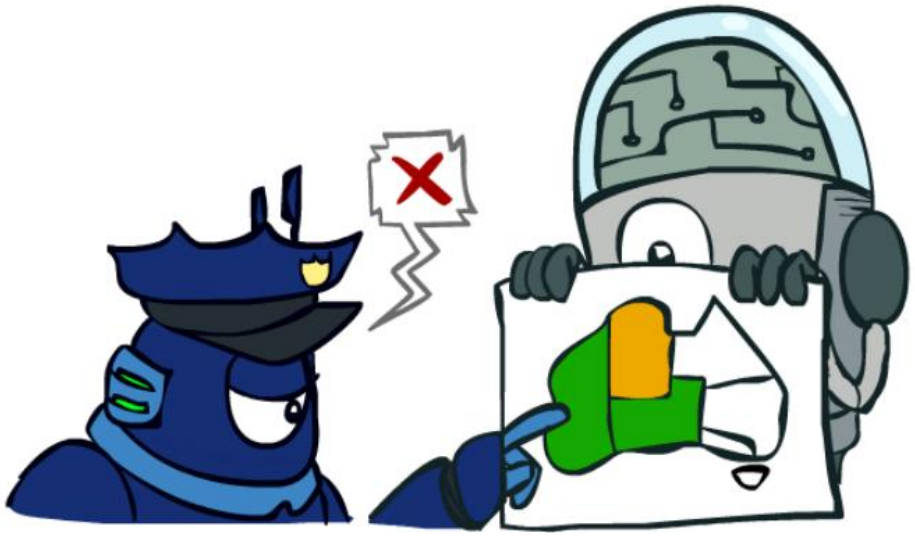
تابع بعدی (Successor) عبارت است از تخصیص یک مقدار به متغیری که هنوز مقدار ندارد.

تابع تست هدف: آیا در حالت فعلی به تمامی متغیرها مقدار داده شده
و آیا این مقادیر تمامی قیود را ارضا می کنند؟

الگوریتم کورکورانه پایه برای مسائل CSP، الگوریتم عقبگرد (Backtracking) می باشد.

جستجوی عقبگرد (Backtracking Search)

الگوریتم کورکورانه پایه برای مسائل CSP، الگوریتم عقبگرد (Backtracking) می باشد.



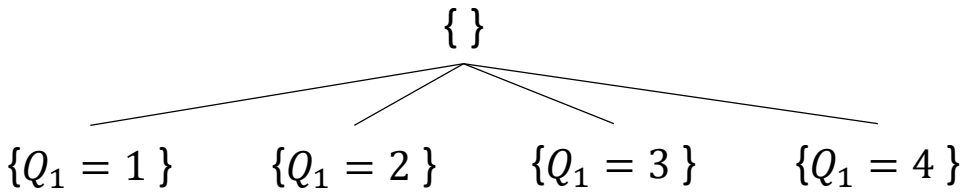
این الگوریتم، نوعی الگوریتم DFS است که در آن

- در هر سطح فقط به یک متغیر مقدار داده می شود.
- در صورتی که در یک حالت قیود نقض نشوند، آن حالت بسط داده می شود و در غیر این صورت بازگشت به عقب (بازگشت به سطح بالاتر اتفاق می افتد.

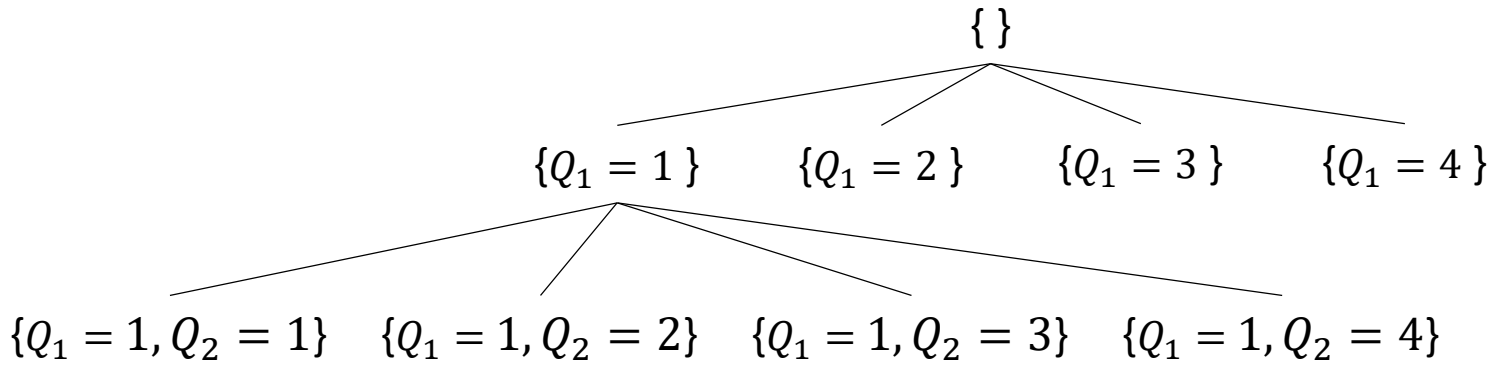
جستجوی عقبگرد (Backtracking Search)

{ }

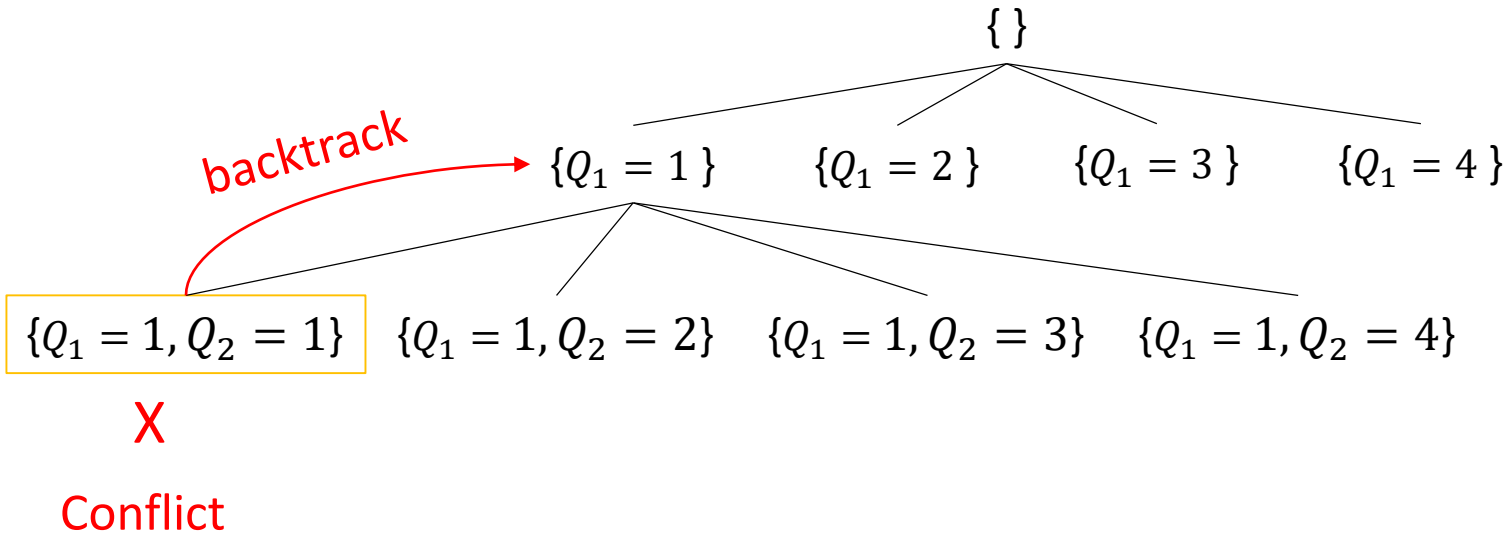
جستجوی عقبگرد (Backtracking Search)



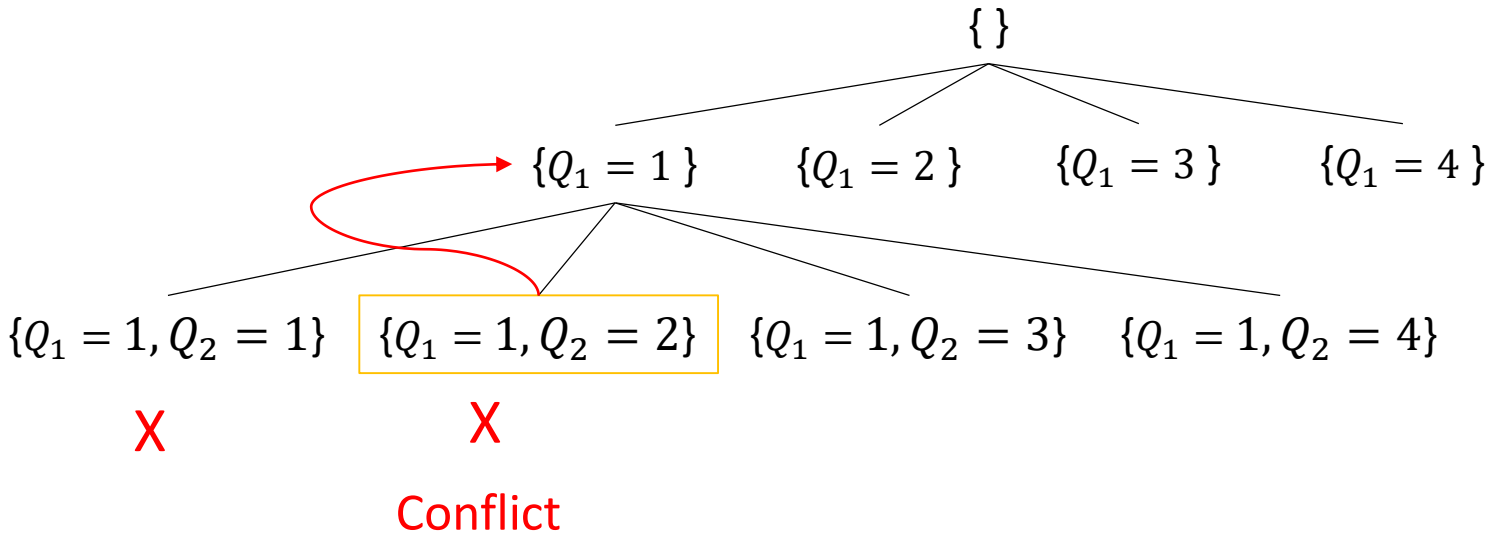
جستجوی عقبگرد (Backtracking Search)



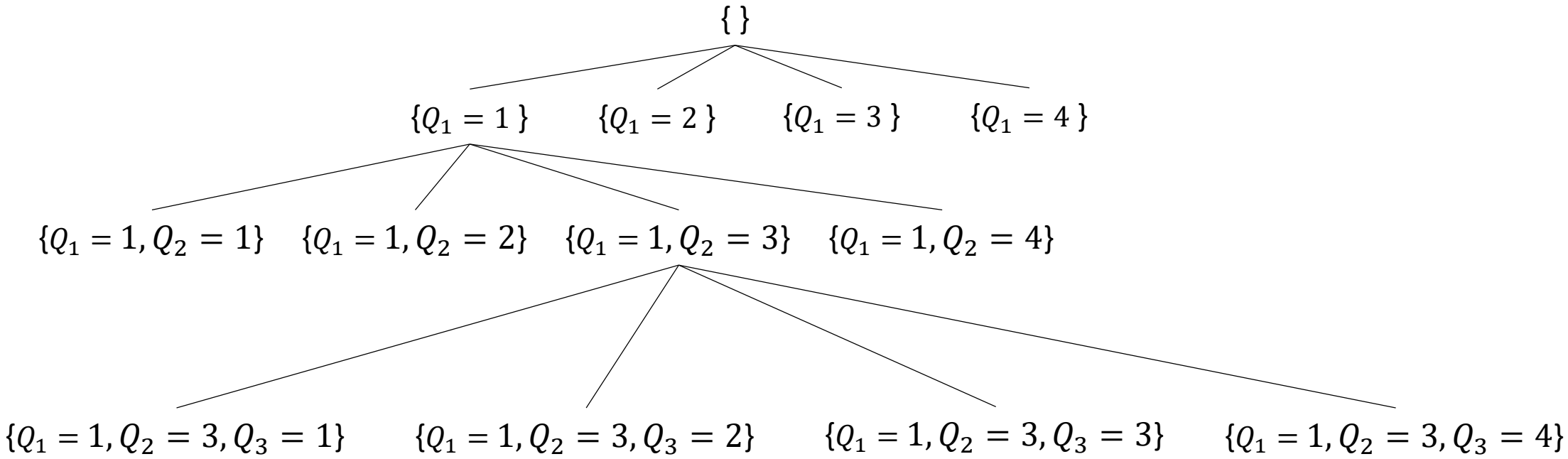
جستجوی عقبگرد (Backtracking Search)



جستجوی عقبگرد (Backtracking Search)



جستجوی عقبگرد (Backtracking Search)



ادامه تمرین

جستجوی عقبگرد (Backtracking Search)

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add {var = value} to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
    return failure
```

جستجوی عقبگرد (Backtracking Search)

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add {var = value} to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
    return failure
```

پیچیدگی زمانی؟ $O(b^n)$

حداکثر عمق درخت برابر با تعداد متغیرها (n) است.

فاکتور انشعاب: تعداد مقادیر ممکن برای هر متغیر (میانگین: b) است.

بهبود جستجوی عقبگرد

فیلترینگ (Filtering)

آیا می توانیم شکستهای احتمالی را پیش از رخداد آن تشخیص دهیم؟

در صورت تشخیص می توان مسیری که به آن شکست می انجامد را بسط نداد.

مرتب سازی (Ordering)

کدام متغیر باید به عنوان تخصیص بعدی انتخاب شود؟

مقادیر این متغیر با چه ترتیبی باید به آن تخصیص داده شوند.

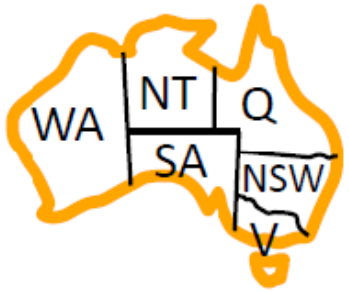
این روش های به ظاهر ساده، باعث بهبود چشمگیری در سرعت اجرای الگوریتم عقبگرد می شوند.

بهبود جستجوی عقبگرد: فیلترینگ

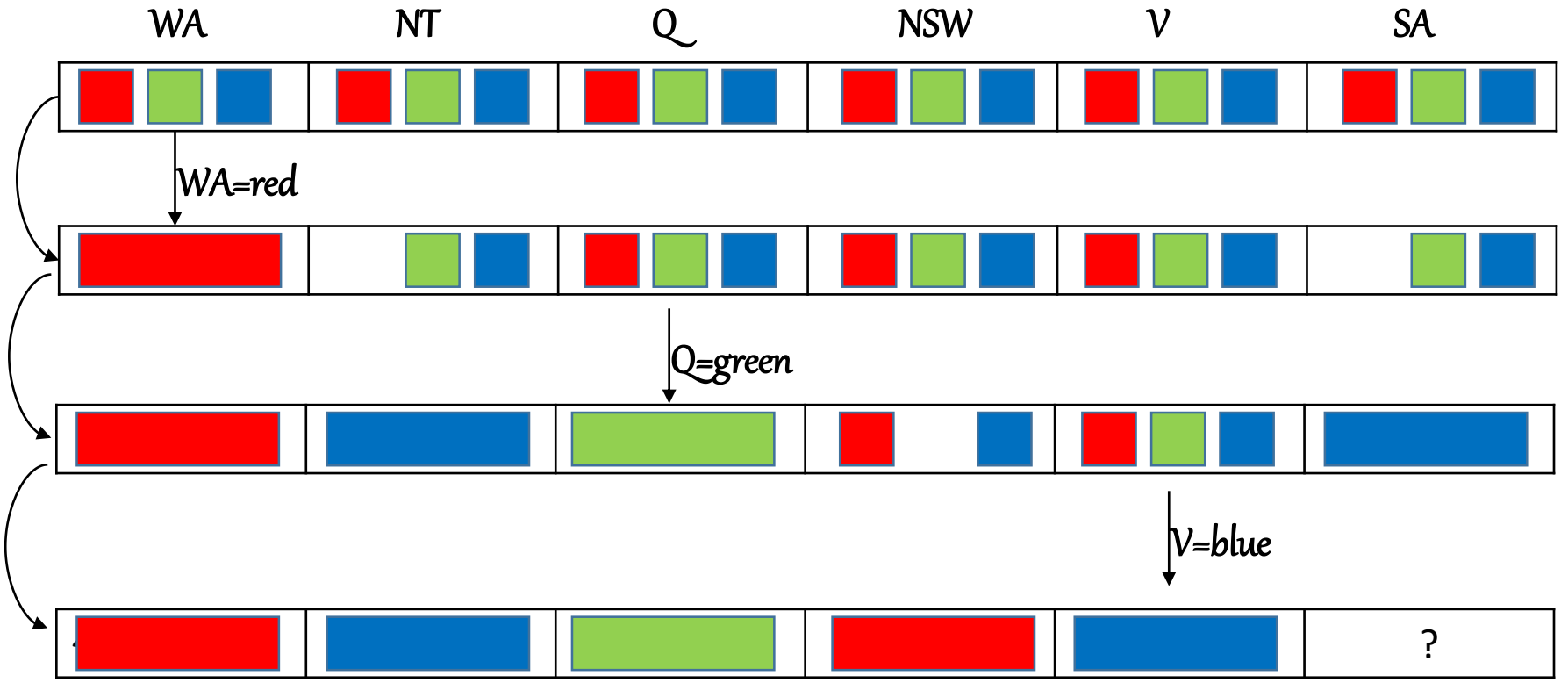
دامنه های متغیرهایی که هنوز مقدار نگرفته اند را بررسی کرده و مقادیر نامناسب را حذف می کنیم.

بهبود جستجوی عقبگرد: فیلترینگ

بررسی رو به جلو (Forward Checking): مقادیری از متغیرهای تخصیص نیافته، که در صورت اضافه شدن به تخصیص فعلی، یک قید را نقض می کنند، حذف کن.



مثال:



با توجه به این که دامنه متغیر SA تهی شده، دیر یا زود این مسیر به شکست منجر خواهد شد. بنابراین، همین حالا می توان عقبگرد کرد.

بهبود جستجوی عقبگرد: مرتب سازی

کدام متغیر باید به عنوان تخصیص بعدی انتخاب شود؟

قانون: همواره متغیری را برای مقداردهی انتخاب کن که کمترین مقدار باقی مانده در دامنه را داشته باشد.

به این قانون MRV (Minimum Remaining Value) گفته می شود.

برای متغیر انتخاب شده، کدام مقدار باید ابتدا تخصیص داده شود؟

قانون: همواره مقداری را انتخاب کن که باعث کمترین تناقض (Conflict) در متغیرهای دیگر شود.

به این قانون LCV (Least Constraining Value) گفته می شود.

بهبود جستجوی عقبگرد

Backtracking +MRV+Forward Checking	Backtracking + MRV	Backtracking+Forward Checking	Backtracking	مسئله
60	(> 1,000K)	2K	(> 1,000K)	P_1
817K	13,500K	(> 40,OOOK)	(> 40,OOOK)	P_2
0.5K	1K	35K	3,859K	P_3

P1: مسئله رنگ آمیزی گراف (۵۰ ایالت آمریکا با ٤ رنگ)

P2: کلیه مسائل n وزیر برای $2 \leq n \leq 50$

P3: مسئله پازل گورخر (Zebra Puzzle)

جلسه آینده: الگوریتم های بهبود سازی تکراری