



سؤال ۱. Binarization of CSP

- الف) برای این که یک محدودیت سه گانه تعریف کنیم، سه متغیر A ، B و C را به شکل زیر تعریف می کنیم:

$$A + B = C$$

یک متغیر جدید به نام AB تعریف می کنیم. اگر دامنه A و B مجموعه اعداد N باشد، آنگاه دامنه AB ، مجموعه $N \times N$ خواهد بود. حال سه محدودیت دو گانه داریم:

۱. یکی بین A و AB که بیان گر این است که مقدار A باید برابر با اولین عضو دوتایی AB باشد.
۲. یکی بین B و AB که بیان گر این است که مقدار B باید برابر با دومین عضو دوتایی AB باشد.
۳. در نهایت یکی که بیان گر این است که جمع دو عضو باید برابر با مقدار C باشد.

همان طور که نشان داده شد، توانستیم یک محدودیت سه گانه را به محدودیت دو گانه تبدیل کنیم. هم چنین می توانیم یک محدودیت چهار گانه متغیرهای A ، B ، C و D را کاهش دهیم. ابتدا باید مراحل بالا را برای A ، B و C انجام دهیم تا محدودیت های دو گانه ایجاد شود و سپس با دوباره اضافه کردن D یک محدودیت سه گانه جدید ایجاد می شود که همانند فرآیندهای بالا قابل تبدیل به محدودیت دو گانه است.

به همین ترتیب با استقرا می توان نتیجه گرفت که هر محدودیت n گانه را می توان به محدودیت $(n - 1)$ گانه تبدیل کرد. نکته: می توان در مرحله محدودیت دو گانه توقف کرد. زیرا هر محدودیت یگانه را می توان به راحتی با حذف کردن آن از دامنه متغیر اعمال کرد.

- ب) چون متغیر D ، محدودیت یگانه دارد و مقدار آن از پیش تعیین شده، کاری به آن نداریم و باید به متغیرها و دامنه های زیر توجه کرده و با توجه به توضیحات داده شده در قسمت الف) عمل کنیم:

$$A \in \{1, 2, 5\}$$

$$B \in \{1, 4, 5, 6, 7\}$$

$$C \in \{10, 12\}$$

$$A + B = C$$

$$A < B$$

یک متغیر جدید به نام AB در نظر می گیریم که عضو اول آن از دامنه متغیر A و عضو دوم آن از دامنه متغیر B است. حال با ضرب دکارتی دامنه A در دامنه B داریم:

$$AB \in \{(1, 1, 10), (1, 1, 12), (1, 4, 10), (1, 4, 12), (1, 5, 10), (1, 5, 12), (1, 6, 10), (1, 6, 12), (1, 7, 10), (1, 7, 12), (2, 1, 10), (2, 1, 12), (2, 4, 10), (2, 4, 12), (2, 5, 10), (2, 5, 12), (2, 6, 10), (2, 6, 12), (2, 7, 10), (2, 7, 12), (5, 1, 10), (5, 1, 12), (5, 4, 10), (5, 4, 12), (5, 5, 10), (5, 5, 12), (5, 6, 10), (5, 6, 12), (5, 7, 10), (5, 7, 12)\}$$

حال با اعمال محدودیت های دو گانه به مجموعه جواب زیر می رسیم:

$$AB \in \{(5, 7, 12)\}, D = 11$$

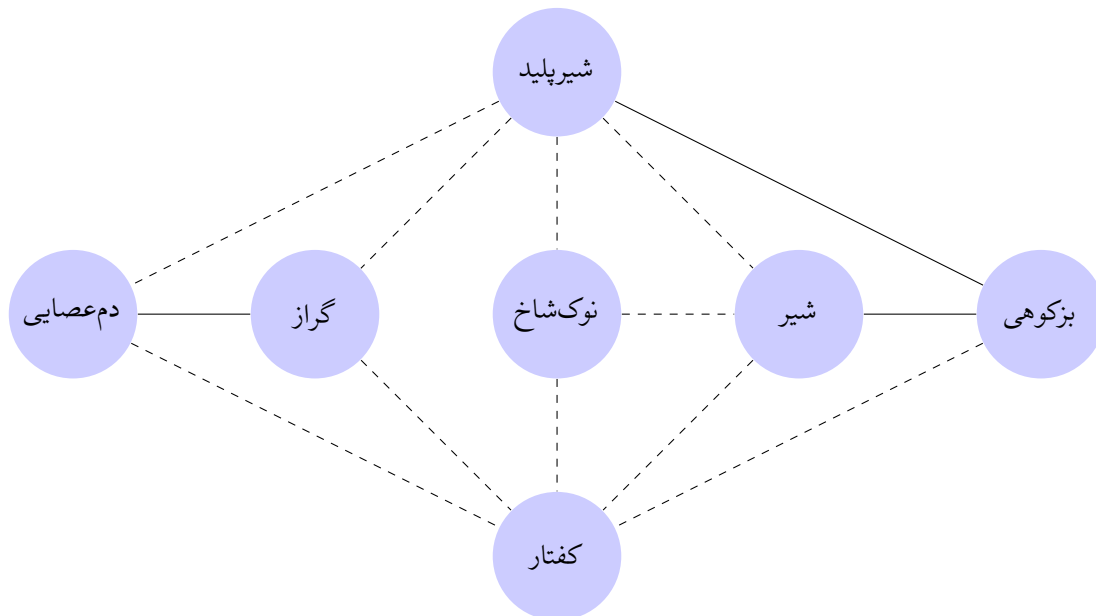
سؤال ٢ . CSP

● (الف)

۱. متغیرها = { شیر، شیر پلید، دم‌عصایی، گراز، کفتار، نوک‌شاخ، بزکوهی }

۲. دامنه هر کدام $\{ ۱, ۲, ۳, ۴ \}$

نکته: این دامنه بدون در نظر گرفتن محدودیت‌هایی است که در صورت سوال ذکر شده است. جلوتر به دامنه‌ی هر کدام خواهیم رسید.



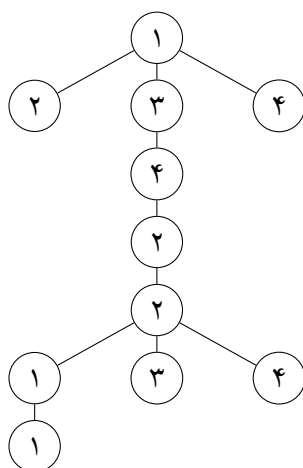
شکل ۱: نقطه چین بیانگر این است که نباید در یک خانه قرار بگیرند. خط هم بیانگر این است که یا حتما باید در یک خانه قرار بگیرند (گراز و دم‌عصایی) و یا نمی‌توانند همسایه یا در یک خانه قرار گیرند (شیر- بزکوهی و بزکوهی- شیرپلید)

دامنه	حیوان
۱	شیر
۴ ۳ ۲	نوک شاخ
۴ ۳	بزکوهی
۴ ۳ ۲	شیر پلید
۴ ۳ ۲	کفتار
۴ ۳ ۲ ۱	دم عسائی
۴ ۳ ۲ ۱	گراز

(ب •

مرحله	متغیر	مقادیر حذف شده همسایه	بک‌ترک
۱	شیر = ۱	–	–
۲	نوکشاخ = ۲	شیرپلید \neq ۲ و گفتار \neq ۲	–
۳	شیر پلید	بزکوهی \neq ۳ و ۴	✓
۴	نوکشاخ = ۳	شیرپلید \neq ۳ و گفتار \neq ۳	–

مرحله	متغیر	مقادیر حذف شده همسایه	بک‌ترک
۵	شیر پلید	بزکوهی $\neq ۳$	-
۶	کفتار	-	-
۷	بزکوهی	شیر پلید $\neq ۴$ و کفتار $\neq ۴$	-
۸	شیر پلید	دم‌عصایی $\neq ۲$ و گراز $\neq ۲$	-
۹	کفتار	-	-
۱۰	گراز	-	-
۱۱	دم‌عصایی	-	-
۱۲	بزکوهی = ۴	-	-
۱۳	شیر پلید = ۲	-	-
۱۴	کفتار = ۲	-	-
۱۵	دم‌عصایی = ۱	گراز $\neq ۳$ و ۴	-
۱۶	گراز	-	-
۱۷	گراز = ۱	-	-



شکل ۲: سطح اول: شیر، سطح دوم: نوک شاخ، سطح سوم: بزکوهی، سطح چهارم: شیر پلید، سطح پنجم: کفتار، سطح ششم: دم‌عصایی و سطح هفتم: گراز

سؤال ۳. CSP

• الف)

۱. متغیرها: $\{ \text{کلاس ۱، کلاس ۲، کلاس ۳، کلاس ۴، کلاس ۵} \}$

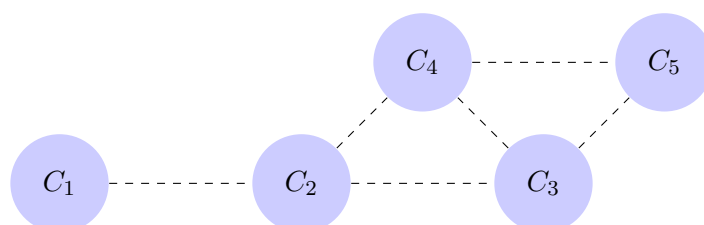
۲. دامنه‌ها:

$$\begin{aligned} C_1 &\in \{C\} \\ C_2 &\in \{B, C\} \\ C_3 &\in \{A, B, C\} \\ C_4 &\in \{A, B, C\} \\ C_5 &\in \{B, C\} \end{aligned}$$

۳. محدودیت‌های دوگانه:

$$C_1 \neq C_2, C_2 \neq C_3, C_2 \neq C_4, C_3 \neq C_4, C_4 \neq C_5, C_3 \neq C_5$$

۴. گراف محدودیت‌ها:



شکل ۳: نقطه‌چین بیان‌گر این است که نباید به‌طور هم‌زمان انجام شوند.

• ب) دامنه متغیرها پس از اعمال lrarc-consistency برابر با حالت زیر خواهد بود:

$$\begin{aligned} C_1 &\in \{C\} \\ C_2 &\in \{B\} \\ C_3 &\in \{A, C\} \\ C_4 &\in \{A, C\} \\ C_5 &\in \{B, C\} \end{aligned}$$

با توجه به این که برای کلاس ۱ فقط پرفسور C (محدودیت یگانه) وجود دارد، با اعمال آن همه‌ی کلاس‌های دیگر به محدودیت یگانه تبدیل می‌شوند و جواب برابر حالت زیر است:

$$\begin{aligned} C_1 &= \text{Professor}(C) \\ C_2 &= \text{Professor}(B) \\ C_3 &= \text{Professor}(A) \\ C_4 &= \text{Professor}(C) \\ C_5 &= \text{Professor}(B) \end{aligned}$$

- الف) توضیح الگوریتم: یک متغیر در مسئله CSP، Arc Consistent است، اگر برای هر مقدار درون دامنه اش، همه ی محدودیت های دوگانه را ارضا کند. معروف ترین الگوریتم برای Arc Consistency الگوریتم AC-3 است. برای این که همه ی متغیرها را Arc Consistent کند، این الگوریتم یک مجموعه از arc ها را در نظر می گیرد^۲. ابتدا این مجموعه شامل همه ی arc های مسئله CSP است. سپس الگوریتم یک arc مانند (X_i, X_j) به طور دل خواه pop می کند و X_i را با توجه به X_j ، consistent می کند. اگر در این مرحله دامنه ی D_i بدون تغییر بماند، الگوریتم سراغ arc بعدی می رود. در غیر این صورت، همه ی arc های (X_k, X_i) را اضافه می کند (که X_k هم سایه ی X_i است). حتی اگر X_k را قبلا در نظر گرفته ایم، باید این کار را انجام دهیم، زیرا ممکن است تغییرات در دامنه ی D_i باعث تغییر (کاهش) در دامنه ی D_k شود. اگر دامنه ی D_i تهی شود، به این معنی است که جواب consistent برای این مسئله وجود ندارد و الگوریتم خطا برمی گرداند. در غیر این صورت، این کار را آن قدر ادامه می دهیم تا به حالتی برسیم که هیچ arc در مجموعه وجود نداشته باشد و به جوابی معادل با جواب CSP اصلی با سرعت بیش تر. می رسم، زیرا دامنه ی متغیرها کوچک تر می شود.

```

function AC-3(csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables  $\{X_1, X_2, \dots, X_n\}$ 
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
     $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\textit{queue})$ 
    if REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) then
      for each  $X_k$  in NEIGHBORS[ $X_i$ ] do
        add  $(X_k, X_i)$  to queue

function REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) returns true iff succeeds
  removed  $\leftarrow$  false
  for each  $x$  in DOMAIN[ $X_i$ ] do
    if no value  $y$  in DOMAIN[ $X_j$ ] allows  $(x, y)$  to satisfy the constraint  $X_i \leftrightarrow X_j$ 
      then delete  $x$  from DOMAIN[ $X_i$ ]; removed  $\leftarrow$  true
  return removed

```

شکل ۴: شبه کد الگوریتم AC-3

بهبود الگوریتم: الگوریتم AC-3 هر یال (X_k, X_i) را هر موقع که مقداری از دامنه ی X_i حذف می شود را در مجموعه قرار می دهد. حتی اگر هر مقدار X_k با چندین مقدار باقی مانده ی مربوط به X_i consistent باشد. فرض کنید برای هر یال (X_k, X_i) ، ما تعداد مقادیر باقی مانده ی مربوط به X_i که با هر مقدار X_k سازگار است را نگهداری می کنیم. ایده ی اساسی این است که محدودیت ها را پیش پردازش کنیم تا برای هر مقدار X_i مقادیری از X_k را نگهداری کنیم که یک یال از X_k به X_i مقداری خاصی از X_i را ارضا کند. این داده ساختار می تواند در زمان متناسب با اندازه مسئله محاسبه شود. سپس، پس از این که مقدار X_i حذف شود، تعداد مقادیر مجاز برای هر یال (X_k, X_i) که در آن ذخیره شده است را یکی کاهش می دهیم. همان طور که مشخص است و توضیح داده شد، زمان اجرای این الگوریتم $O(n^2 d^2)$ است.

- ب) خیر. فرض کنید یک مسئله ی CSP داریم که به شرح زیر است:

$$\text{Variables} = \{X, Y\}$$

$$D_X = D_Y = \{1, 2, 3, 4\}$$

محدودیت های آن هم $X < 4$ و $X = Y$ هستند. اگر چه این مسئله 2-consistent است اما 1-consistent نیست. زیرا به ازای $X = 4$ شرط (محدودیت) اول نقض می شود.

- ج) همان طور که در اسلاید شماره ۳۱ جلسه ۷ و ۸ آمده، اثبات می کنیم که این مسئله جواب دارد.

^۲ در مجموعه ترتیب مهم نیست

اثبات: اگر یک مسئله CSP، Strong n -consistent باشد، به این معنی است که

$$(n-1) - \text{consistent}, (n-2) - \text{consistent}, \dots, 1 - \text{consistent}$$

است. حال می‌توانیم مسئله را به شکل زیر حل کنیم:

۱. یک مقدار consistent برای X_1 انتخاب می‌کنیم.

۲. چون 2-consistent نیز هست، تضمین می‌شود که مقدار consistent برای X_2 نیز پیدا می‌شود.

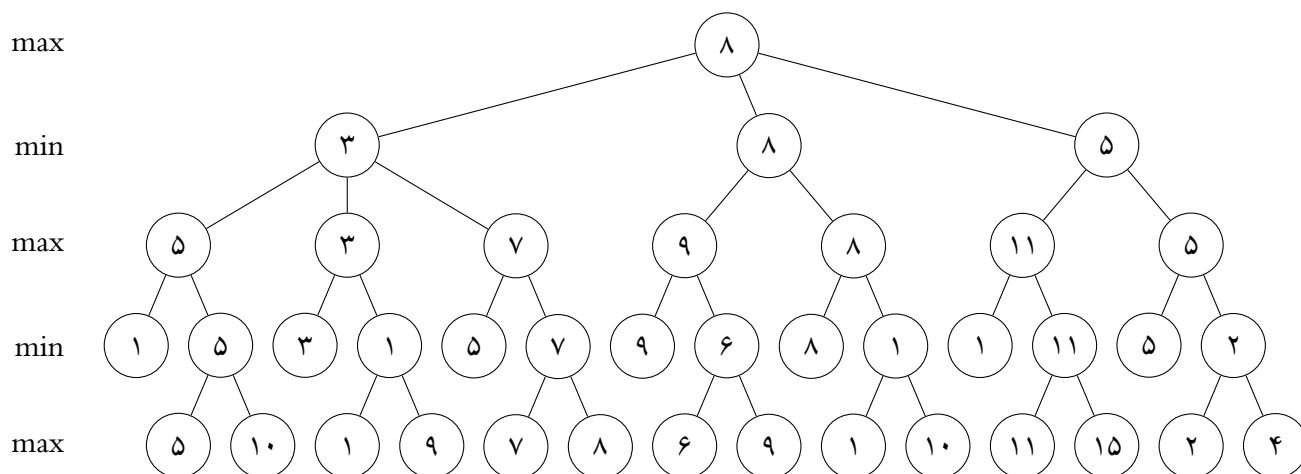
۳. برای X_3 هم همانند قسمت (۲) چون 3-consistent است، تضمین می‌شود که مقدار قابل قبول برای آن پیدا می‌شود.

۴. همین‌طور مراحل ادامه می‌یابد. برای هر متغیر X_i تنها باید داخل دامنه‌ی با سایز d آن به دنبال مقدار consistent با X_1, \dots, X_{i-1} بگردیم.

نکته: زمان اجرای این الگوریتم $O(n^2d)$ است.

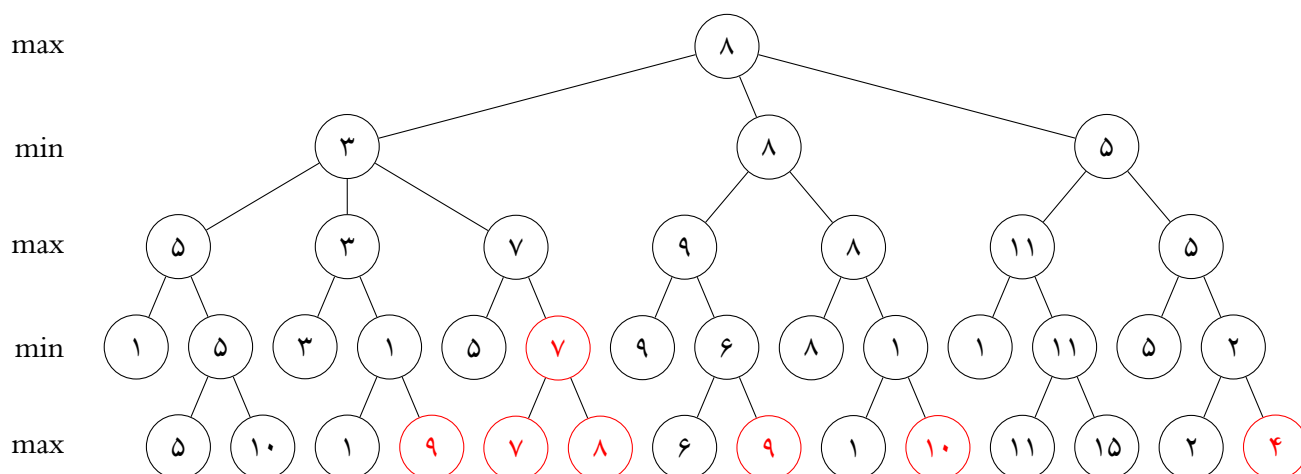
سؤال ۵. Minimax

• الف)



شکل ۵: مقدار ریشه برابر با هشت است.

• ب)



شکل ۶: گره‌هایی که رنگ آن‌ها قرمز است، حذف می‌شوند.

• ج) برای این‌که بتوانیم بیش‌ترین تعداد حذف را در الگوریتم $\alpha - \beta$ pruning داشته باشیم، باید از چپ به راست، عددهای زیر را در برگ قرار دهیم:

9, 8, 8, 10, 9, 9, 15, 10, 11, 5, 5, 7, 6, 7, 1, 1, 1, 4, 2, 3

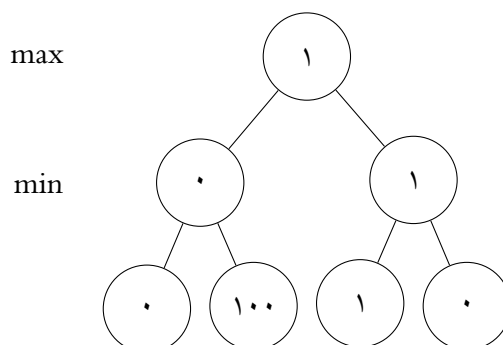
سؤال ۶. Minimax

- الف) حد بالای تعداد گره‌های انتهایی درخت minimax برابر با $n!$ است. حد بالا برای تمامی گره‌ها برابر با $\sum_{i=1}^n i!$ است. این مقدار تفاوت خیلی زیادی (از لحاظ مرتبه‌ای) با $n!$ ندارد.
حد بالا برای تعداد حالت‌های متمایز بازی برابر با 3^n است. چون هر کدام از خانه‌ها یا خالی هستند، یا X و یا O هستند.
- ب) در این حالت هیچ بازی‌ای سریع تمام نمی‌شود. اگر حالت‌های تکراری را در نظر نگیریم دقیقاً $\sum_{i=1}^n i!$ حالت داریم.
در آخر بازی خانه‌ها بین دو بازیکن تقسیم می‌شود. یعنی $\lceil n/2 \rceil$ خانه به بازیکن اول و $\lfloor n/2 \rfloor$ خانه به بازیکن دوم می‌رسد. بنابراین یک حد پایین خوب برای تعداد حالت‌های متمایز همان تعداد حالت‌های انتهایی متمایز است که برابر با $\binom{n}{\lfloor n/2 \rfloor}$ می‌باشد.
- ج) اگر برای حالت s تابع $X(s)$ را تعداد حالات بردی که در آن هیچ O ای وجود ندارد و تابع $O(s)$ را تعداد حالات بردی که در آن هیچ X ای وجود ندارد در نظر بگیریم، تابع زیر می‌تواند یک تابع ارزیابی برای آن به حساب بیاید:

$$evaluation - function(s) = X(s) - O(s)$$

سؤال ۷. Minimax vs Expectimax

• الف)



همان‌طور که می‌بینیم در درخت بالا اگر از الگوریتم minimax استفاده کنیم، جوابی برابر با ۱ خواهیم داشت. اما اگر از الگوریتم expectimax استفاده کنیم مقدار ۵۰ را می‌گیریم.

اگر بازیکن اول انتظار داشته باشد که بازیکن دوم به طور تصادفی حرکت کند، بهتر است از الگوریتم expectimax استفاده کند.

• ب) طبق تعریف حرکت بهینه کمینه به معنی حرکتی است که کوچک‌ترین مقدار ممکن را می‌گیرد. بازی تصادفی شامل حرکاتی است که لزوماً بهینه نیستند. با فرض این که هیچ دو tie با یک‌دیگر برابر نیستند، می‌توان گفت الگوریتم expectimax یک الگوریتم زیربهینه^۳ است. میانگین‌گیری در برابر کمینه‌گیری خروجی را افزایش می‌دهد.

با این حساب، می‌توانیم ثابت کنیم که درخت بازی‌ای وجود ندارد که در آن مقدار ریشه برای آن برای expectimax کوچک‌تر از minimax باشد. یکی از آن‌ها بازی بهینه است و دیگری بازی زیربهینه است.

اگر بازیکن دوم به طور بهینه بازی کند بازیکن اول باید از الگوریتم minimax استفاده کند.