سوال اول)

الف) نادرست: پیچیدگی زمانی الگوریتم بک‌ترک، هنگامی که روی گرافی که دور داشته باشد اعمال شود از $O(d^n)$ است. با وجود هیوریستیک‌های مختلف نظیر LCV و MRV نیز نمی‌توان پیچیدگی زمانی را به $O(nd^2)$ رساند. فرض کنید که کاردینالیتی دامنه متغیرها به اندازه‌ی کافی بزرگ باشد. این هیوریستیک‌ها عملا تاثیری روی عملکرد الگوریتم نخواهند داشت زیرا به طور کلی نمی‌توانند حالت‌های زیادی را حذف کنند.

ب) نادرست: طبق مطالب گفته شده در کلاس، اگر گراف محدودیت‌ها در یک مسئله‌ی CSP یک درخت باشد، کافی‌است به ازای تمامی یال‌ها، d^2 حالت را برای متغیرهای دو طرف یال چک کنیم و پیچیدگی زمانی از $O(nd^2)$ می‌شود و نه از $O(n^2)$.

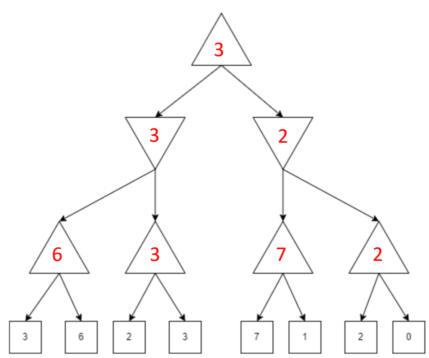
ج) نادرست: الگوریتم هرس آلفا بتا، زیردرخت‌هایی از گراف مینیمکس را هرس می‌کند که قطعا تاثیری روی مقدار راس پدر پدر آن زیر درخت نخواهند داشت. در نتیجه مقدار بدست آمده برای ریشه‌ی درخت مینیمکس هنگام استفاده از الگوریتم هرس تغییر نخواهد کرد.

د) درست: اگر مقادیر فرزندان یک راس از درخت مینیمکس به صورت x_1 تا x_n بصورت صعودی باشند، با اعمال تابع اکید صعودی f روی این مقادیر، مقادیر $f(x_1)$ تا $f(x_n)$ نیز صعودی خواهد بود و ترتیب‌شان تغییر نخواهد کرد. با توجه به اینکه مقدار پدر این فرزندان یکی از مقادیر x_1 یا x_n قبل از اعمال تابع f بوده‌است، پس از اعمال این تابع نیز همان زیردرخت بعنوان مقدار min یا max انتخاب خواهد شد. در نتیجه در نهایت، مسیر انتخاب شده در هر راس تفاوتی نخواهد داشت.

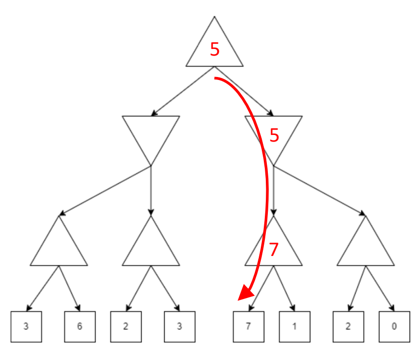
هـ) درست: فرض کنید می‌خواهیم مقدار راس n را در درخت مینیمکس مشخص کنیم. فرزندان این راس نیز با S_1 تا S_k مشخص می‌کنیم. فرض کنید مقادیر S_1 تا S_i تا به اینجا پیدا شده‌است. هنگام پیدا کردن مقدار فرزند S_{i+1}، بسته به اینکه راس n راس max/min باشد، اگر مقدار یکی از فرزندان S_{i+1} کمتر یا بیشتر از مقادیر بدست آمده برای S_1 تا S_i باشد، باقی فرزندان S_{i+1} بررسی نمی‌شوند و به اصطلاح هرس می‌شوند. با توجه به اینکه اعمال تابع f روی طبقه‌ای که پایین طبقه‌ی فرزندان راس n است تاثیری روی ترتیب مقدار این رئوس نخواهد داشت، در نتیجه، برگ‌هایی که توسط این الگوریتم هرس می‌شوند هیچ تغییری نخواهند کرد.

سوال دوم)

الف)



ب) بیشترین امتیازی که بازیکن اول می‌تواند کسب کند، مقدار 7-c است. این کار را می‌تواند با مجبور کردن بازیکن دوم به انتخاب مسیری که منجر می‌شود بازیکن اول مقدار ۷ را انتخاب کند بدست آورد. اگر هزینه‌ی c را پرداخت نکند، می‌تواند مطمئن باشد که امتیاز ۳ را بدست می‌آورد. با توجه به اینکه در این بخش مقدار c=2 است، پس به صرفه است که بازیکن اول از قابلیتش استفاده کند و مسیر زیر را برای بیشترین امتیاز ۵ را پیش برود:



ج) اما برای این بخش، چون مقدار 2 = 7-c می‌شود که از حداقل امتیاز بدست آمده بدون استفاده از قابلیت کمتر است، برای بازیکن اول نمی‌صرفد که از قابلیتش استفاده کند.

سوال سوم)

الف) برای داشتن یک مسئله‌ی CSP نیاز است که متغیرها، دامنه‌ی متغیرها و قیود را تعریف کنیم.

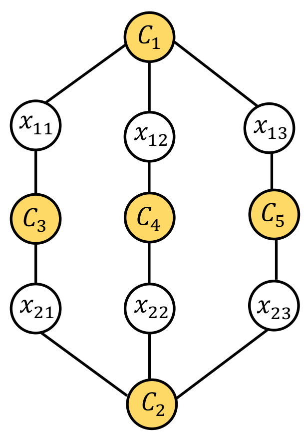
- متغیرها را به ترتیب x_{11} تا x_{23} می‌نامیم.
- دامنه‌ی متغیرهای تعریف شده در زیر آمده‌است:

$D(x_{11}) = \{4,5\}$
$D(x_{12}) = \{5,6\}$
$D(x_{13}) = \{3,4\}$
$D(x_{21}) = \{1,2\}$
$D(x_{22}) = \{1,2\}$
$D(x_{23}) = \{3,4\}$

- قیودی که روی متغیرها داریم نیز به ترتیب C_1 تا C_2 می‌نامیم. قیود را به صورت زیر تعریف می‌کنیم:

$C_1: x_{11} + x_{12} + x_{13} = 14$
$C_2: x_{21} + x_{22} + x_{23} = 7$
$C_3: x_{11} + x_{21} = 7$
$C_4: x_{12} + x_{22} = 6$
$C_5: x_{13} + x_{23} = 8$

ب) با توجه به اینکه قیود، فقط قیود باینری نیستند، باید یک hypergraph طراحی کنیم که گرافی دو بخشی شامل راس‌های قید و راس‌های متغیر باشد. شکل گراف در زیر آمده‌است:



ج) استیت‌های مختلف بک‌ترک در زیر آمده‌است:

0) $x_{11} = \{4,5\},\ x_{12} = \{5,6\},\ x_{13} = \{3,4\},\ x_{21} = \{1,2\},\ x_{22} = \{1,2\},\ x_{23} = \{3,4\}$

- Same remaining values, Same degrees → Random pick

1) $x_{11} = 4,\ x_{12} = \{5,6\},\ x_{13} = \{3,4\},\ x_{21} = -,\ x_{22} = \{1,2\},\ x_{23} = \{3,4\}$

- Forward checking, no value for x_{21} → Backtrack

2) $x_{11} = 5,\ x_{12} = \{5,6\},\ x_{13} = \{3,4\},\ x_{21} = \{2\},\ x_{22} = \{1,2\},\ x_{23} = \{3,4\}$

- x_{21} is minimum remaining value → pick x_{21}

3) $x_{11} = 5,\ x_{12} = \{5,6\},\ x_{13} = \{3,4\},\ x_{21} = 2,\ x_{22} = \{1,2\},\ x_{23} = \{3,4\}$

- Same remaining values, Same degrees → Random pick

4) $x_{11} = 5,\ x_{12} = 5,\ x_{13} = \{4\},\ x_{21} = 2,\ x_{22} = \{2\},\ x_{23} = \{3,4\}$

- x_{22} and x_{13} have same remaining values → Random pick between the two

5) $x_{11} = 5,\ x_{12} = 5,\ x_{13} = 4,\ x_{21} = 2,\ x_{22} = \{1\},\ x_{23} = \{4\}$

- x_{22} and x_{23} have same remaining values → Random pick between the two

6) $x_{11} = 5,\ x_{12} = 5,\ x_{13} = 4,\ x_{21} = 2,\ x_{22} = 1,\ x_{23} = \{4\}$

- Only one variable remaining → Pick it!

7) $x_{11} = 5,\ x_{12} = 5,\ x_{13} = 4,\ x_{21} = 2,\ x_{22} = 1,\ x_{23} = 4$

- Finished!

سوال چهارم)

الف)

۱) محدب است، از این قضیه استفاده می‌کنیم که اگر مشتق دوم تابعی همواره مثبت باشد، آن تابع محدب است.

$\forall x : f''(x), g''(x) \geq 0$

$h''(x) = f''(x) + tg''(x) \xrightarrow{t\geq 0} h''(x) \geq 0$

۲) محدب است

$k(\theta x + (1-\theta)y) = \max\{f(\theta x + (1-\theta)y), g(\theta x + (1-\theta)y)\}$

$\leq \max\{\theta f(x) + (1-\theta)f(y), \theta g(x) + (1-\theta)g(y)\}\ (\mathbf{1})$

$\leq \theta \max\{f(x), g(x)\} + (1-\theta)\max\{f(y), g(y)\}\ (\mathbf{2})$

$= \theta k(x) + (1-\theta)k(y)$

نامساوی **(۱)** از اینجا می‌آید توابع f محدب است و مقدار $f(\theta x + (1-\theta)y)$ همواره از $\theta f(x) + (1-\theta)f(y)$ کوچک‌تر است. به همین ترتیب برای تابع g نیز داریم. در نتیجه $\max\{f(\theta x + (1-\theta)y), g(\theta x + (1-\theta)y)\}$ همواره از $\max\{\theta f(x) + (1-\theta)f(y), \theta g(x) + (1-\theta)g(y)\}$ که هر دو آرگومان تابع max دوم از هر دو آرگومان تابع max اول بزرگتر هستند، بزرگتر است.

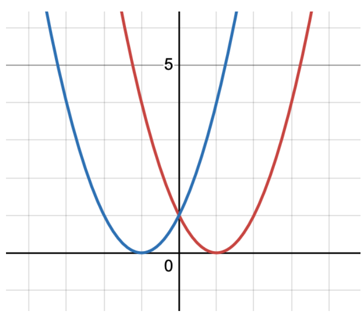
نامساوی **(۲)** از اینجا می‌آید که مقدار $\max\{\theta f(x) + (1-\theta)f(y), \theta g(x) + (1-\theta)g(y)\}$، یکی از دو مقدار $\theta f(x) + (1-\theta)f(y)$ و یا $\theta g(x) + (1-\theta)g(y)$ می‌باشد. $\theta \max\{f(x), g(x)\} + (1-\theta)\max\{f(y), g(y)\}$ از هر دو این مقدارها بزرگتر است.

۳) محدب نیست، برای اثبات مثال نقض زیر را می‌آوریم:

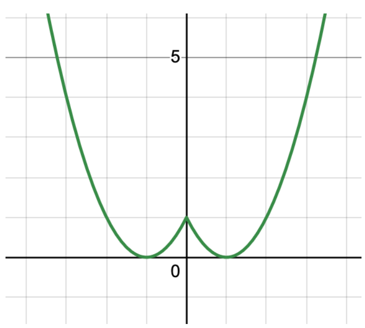
فرض کنید دو سهمی داریم که شکل‌های زیر را در صفحه کارتزین را دارند.

$f(x) = (x-1)^2$

$g(x) = (x+1)^2$



مقدار $\min\{f(x), g(x)\}$ شکل زیر را به خود می‌گیرد، که واضحا محدب نیست:



۴) محدب نیست، برای اثبات مثال نقض زیر را می‌آوریم:

از این قضیه استفاده می‌کنیم که اگر مشتق دوم تابعی همواره مثبت باشد، آن تابع محدب است.

$$f(x) = (x-1)^2$$
$$g(x) = (x+1)^2$$
$$s''(x) = \left(f(x)g(x)\right)'' = f''(x) + g''(x) + 2f'(x)g'(x) = 4 + 8(x-1)(x+1)$$
$$s''(x=0) = -4 < 0 \ (not\ convex)$$

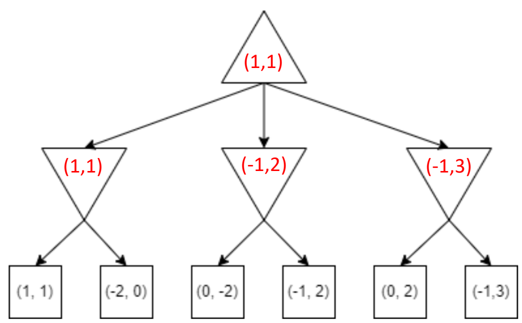
ب) محدب است،

دو عضو دلخواه از این مجموعه انتخاب می‌کنیم.

$$P_1\,, P_2 \in C$$
$$||x_1|| \le t_1, ||x_2|| \le t_2$$
$$\theta P_1 + (1-\theta)P_2 = (\theta x_1 + (1-\theta)x_2, \theta t_1 + (1-\theta)t_2)$$
$$||\theta x_1 + (1-\theta)x_2|| \le ||\theta x_1|| + ||(1-\theta)x_2|| = \theta||x_1|| + (1-\theta)||x_2|| \le \theta t_1 + (1-\theta)t_2$$
$$\rightarrow\ ||\theta x_1 + (1-\theta)x_2|| \le \theta t_1 + (1-\theta)t_2 \rightarrow \theta P_1 + (1-\theta)P_2 \in C\ (convex\ set)$$

سوال پنجم

الف)



ب)

در بازی‌های non zero-sum هر بازیکن به دنبال بیشینه کردن امتیاز خود است و امتیاز حریف کاملا از امتیاز شما مستقل است. برای همین نمی‌توان زیردرختی را قبل از مشاهده‌ی همه‌ی زیر درخت‌های یک راس حذف کرد. اگر برای همه‌ی رئوس $U_a = U_b$ باشد، در این صورت مسئله به پیدا کردن بیشتر مقدار است که قبل از پیمایش کل درخت minimax، نمی‌توان با قطعیت آن را پیدا کرد.

ج)

خیر، درخت کامل‌شده‌ی minimax با این فرض کامل شده که جفت بازیکن‌ها به دنبال بیشینه کردن امتیاز خود هستند. در همین مثال، اگر حریف ما به جای بیشینه کردن امتیاز خود به دنبال کمینه کردن امتیاز ما باشد، امتیاز ما حداکثر 1- می‌شود که نشان می‌دهد مقدار راس درخت minimax در بازی‌های non zero-sum، مقدار worst-case را معرفی نمی‌کند.

د و ه)

در الگوریتم هرس آلفابتا، هرس در راس‌های min در این زمان رخ می‌دهد: به فرزندی برسیم که مقدار آن از کمینه مقدار راس‌های دیگر کوچکتر باشد. زیرا در این صورت راس max که پدر راس min است قطعا آن راس را انتخاب نمی‌کند. حال در بازی‌های nearly zero-sum، می‌خواهیم وقتی هرس انجام دهیم که مطمئن هستیم مقدار راس min برای پدرش که راس max است بی‌اهمیت است. داریم:

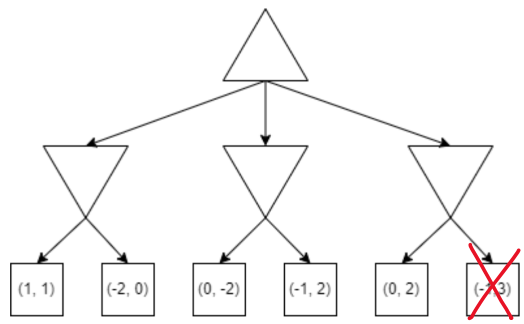
$$U_A(S) + U_B(S) \le \epsilon$$
$$\alpha \rightarrow maximum\ value\ found$$

$U_B(S) > \epsilon - \alpha \rightarrow U_A(S) < \alpha$

$proof: \epsilon - \alpha + U_A(S) < U_A(S) + U_B(S) \leq \epsilon \rightarrow U_A(S) < \alpha$

پس اگر در فرزندان یک راس min، فرزندی یافتیم که $U_B(S) > \epsilon - \alpha$، می‌توان مطمئن بود که $U_A(S) < \alpha$ و می‌توان آن زیردرخت را هرس کرد.

این شرط عمومی، شرط کافی‌است که راسی را هرس کنیم. حال این شرط را برای درخت minimax سوال اعمال می‌کنیم:



راس min چپ برگ چپ: مقدار $\alpha = 1$ بدست می‌آید و مقدار (1,1) برای این راس انتخاب می‌شود.

برگ راست: چون مقدار $U_B(S) = 1 \leq 1$، در نتیجه این برگ هرس نمی‌شود.

راس min وسط برگ چپ: (0,-2) بعنوان راس انتخاب می‌شود. در اینجا مقدار $U_B(S) = -2 \leq 1$ پس باید برگ بعدی را بررسی کنیم.

برگ راست: مقدار (-1,2) بعنوان مقدار راس انتخاب می‌شود.

راس min راست برگ چپ: در اینجا مقدار (0,2) بعنوان مقدار راس انتخاب می‌شود. در اینجا مقدار $U_B(S) = 2 > 1$، برگ‌های بعدی این راس هرس می‌شوند.

برگ راست: هرس شد!

تنها راست ترین برگ (-1,3) هرس می‌شود.

و)

در حالت اول مقدار ریشه بصورت (U_A,U_B) است. در حالت دوم مقدار ریشه بصورت (V_A,V_B) است و چون بازیکن دوم بهینه نکرده است داریم $V_B \leq U_B$

معادلات زیر را داریم:

$|U_A + U_B| \leq \epsilon \rightarrow U_B \leq \epsilon - U_A$

$|V_A + V_B| \leq \epsilon \rightarrow V_B \geq -\epsilon - V_A$

$-\epsilon - V_A \leq V_B \leq U_B \leq \epsilon - U_A$

$U_A - 2\epsilon \leq V_A$

در نتیجه، در بدترین حالت، مقدار $V_A = U_A - 2\epsilon$ است.

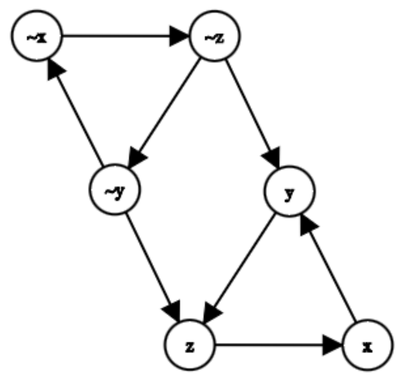
سوال ششم)

الف)

در این مسئله، n متغیر داریم که آن‌ها را با x_1 تا x_n نمایش می‌دهیم. هر کدام از این متغیرها، دارای دامنه D={0,1} هستند. هرکدام از قیدها را نیز به صورت or دو تا از متغیرها نمایش می‌دهیم. یعنی به ازای قید C_k داریم:

$$\forall k, C_k = x_i \vee x_j = 1, \quad 0 \le\ i, j < n$$

گراف مد نظر برای مسئله نامبرده به صورت زیر است:



حل مسئله: x = y = z = True (1)

ب)

مولفه‌های قویا همبند را در این گراف می‌توان به این صورت نوشت: اعضای یک مولفه‌ی قویا همبند با یکدیگر هم‌ارز هستند. زیرا هر یال جهت‌دار در این گراف به صورت p می‌دهد q قرار داده شده‌است و اگر دو عضو در یک مولفه قویا همبند باشند، این یعنی هر دو به یکدیگر مسیر دارند. یعنی برای هر دو راس p و q در یک مولفه قویا همبند داریم:

$$(p \to p_1 \to \cdots \to q) \wedge (q \to q_1 \to \cdots \to p) \equiv p\ \leftrightarrow q$$

اگر یک عنصر و نقیض آن در یک مولفه قویا همبند باشند، داریم:

$$(x\ \leftrightarrow \sim x) \to\ contradiction$$

پس اگر در یک مولفه قویا همبند یک عنصر و نقیض آن موجود باشند، قطعا جوابی برای مسئله نخواهیم داشت.

برای اینکه عکس قضیه را ثابت کنیم، باید الگوریتمی معرفی کنیم که جوابی برای این مسئله پیدا کند. در قسمت بعدی این الگوریتم را معرفی می‌کنیم.

ج)

الگوریتم زیر را معرفی می‌کنیم:

۱) با استفاده از ۲ جست‌وجوی dfs، مولفه‌های قویا همبند این گراف را پیدا می‌کنیم. این کار را با استفاده از الگوریتم Kosaraju انجام می‌دهیم که از O(n+m) است.

۲) به هر مولفه قویا همبند یک اندیس می‌دهیم و سپس، اگر یالی بین دو عضو درون دو مولفه قویا همبند جدا وجود داشت، این دو مولفه را به هم متصل می‌کنیم. به اصطلاح condensation آن گراف را تولید می‌کنیم.

۳) گراف جدید تولید شده را به صورت توپولوژیک سورت می‌کنیم. می‌دانیم که این گراف یک DAG است پس topological sort دارد.

۴) برعکس ترتیب توپولوژیک حرکت می‌کنیم. به ترتیب هر مولفه‌ای که می‌بینیم را اگر اعضای آن مقدار دهی نشده باشند، آن‌ها را برابر 1 قرار می‌دهیم و مقادیر complement این اعضا را برابر false قرار می‌دهیم.

اینگونه در O(n+m) مسئله را حل کردیم.