جواب تکلیف سری سوم

درس نظریه محاسبه - دانشکده ریاضی
دانشگاه صنعتی خواجه نصیرالدین طوسی. بهار ۱۴۰۲

۱. بطور مختصر اثباتی برای گزاره‌های زیر ارائه کنید. فرض کنید ماشین تورینگ دو نواره است. یک نوار فقط خواندنی که ورودی روی آن قرار گرفته است. یک نوار خواندنی‌ـ نوشتنی برای انجام محاسبات.

- $\text{TIME}(t(n)) \subseteq \text{SPACE}(t(n))$

طبیعتا یک ماشین تورینگ به فضای مصرفی بیشتر از زمان اجرایش نیاز ندارد چون در هر قدم حداکثر یک سلول جدید از نوار ملاقات می‌شود. دقت کنید اینجا فرض کرده‌ایم که ماشین تورینگ دو نواره است. نوار رشته ورودی که فقط خواندنی است و نوار اجرایی که برای نوشتن مراحل میانی و جواب نهایی در نظر گرفته شده است.

- $\text{SPACE}(t(n)) \subseteq \text{TIME}(2^{O(t(n))})$

اینجا فرض می‌کنیم $t(n) > \log n$. ماشین تورینگ M با فضای مصرفی $t(n)$ را در نظر بگیرید. گراف پیکربندی configuration graph ماشین M برای ورودی x به تعداد $2^{O(t(n))}$ پیکربندی مختلف دارد (الفبای ورودی و تعداد وضعیتهای ماشین یک عدد ثابت است.) با داشتن توصیف M و رشته ورودی x می‌توانیم گراف پیکربندی $G_{M,x}$ را در زمان $2^{O(t(n))}$ ایجاد کنیم. دقت کنید تعداد یالهای گراف از مرتبه تعداد رئوس است. سپس چک می‌کنیم آیا از پیکربندی شروع به پیکربندی پذیرش مسیری در $G_{M,x}$ وجود دارد یا نه. همه اینها در زمان مذکور قابل انجام است.

۲. زبان زیر را در نظر بگیرید.

$$\text{Non-Bipartite} = \{G| \text{ گراف } G \text{ دوبخشی نیست } \}$$

نشان دهید $\text{Non-Bipartite} \in L^2$

کافی است نشان دهیم $\text{Non-Bipartite} \in \boldsymbol{NL}$ چون با قضیه Savitch می‌دانیم که $\boldsymbol{NL} \subseteq \boldsymbol{L}^2$. می‌دانیم که یک گراف دوبخشی نیست اگر و فقط اگر یک دور به طول فرد داشته باشد. با استفاده از یک الگوریتم غیرقطعی دور به طول فرد را در گراف G حدس می‌زنیم. برای این کار از یک راس دلخواه شروع کرده و راس بعدی را حدس می‌زنیم به تعداد فرد راس جلو می‌رویم اگر به راس اولیه بازگشتیم یعنی دوری به طول فرد در گراف وجود دارد. فقط کافی است مشخصه راس شروع و طول دور را نگه‌داری کنیم. این در فضای $O(\log n)$ قابل انجام است.

۳. نشان دهید زبان UCYCLE عضو کلاس L است.

$$\text{UCYCLE} = \{G| \text{ گراف غیر جهتدار } G \text{ دور ندارد } \}$$

راهنمایی: ببینید چطور می‌توانید با استفاده از پیمایش DFS ، درخت را از غیر درخت تشخیص دهید.

در پیمایش DFS یک درخت، وقتی پیمایش را از راس u شروع می‌کنیم از همان یالی که از u خارج شده‌ایم به u برمی‌گردیم. اما وقتی u در یک دور باشد (گراف درخت نباشد) از یک همسایه متفاوت به u برمی‌گردیم. پس برای تشخیص درخت از گرافهایی که دور دارند، کافی است که همه رئوس را امتحان کنیم. از هر راس گراف یک پیمایش DFS جدید را شروع می‌کنیم. در هر پیمایش DFS کافی است راس شروع u و یالی که از آن خارج شده‌ایم را در حافظه نگهداری کنیم. این در فضای $O(\log n)$ قابل انجام است. اگر راسی پیدا شد که از یالی متفاوت به آن برگشتیم، گراف دور دارد، در غیر اینصورت یک درخت است.

۴. یک سکه سالم را ۱۰ بار پرتاب می‌کنیم. احتمالات زیر را بدست آورید.

(آ) تعداد شیرها از خطها بیشتر باشد

فرض کنید E رخداد تساوی تعداد شیرها و خطها باشد. داریم $Pr[E] = (\frac{1}{2})^5\binom{10}{5}(\frac{1}{2})^5$ چون شیر و خط متقارن هستند، احتمال اینکه تعداد شیرها بیشتر باشد برابر است با

$$\frac{1}{2}(1 - Pr[E])$$

(ب) حداقل ۴ شیر متوالی داشته باشیم.

در 10 پرتاب، فرض کنید E رخداد آمدن 4 شیر متوالی باشد. حال فرض کنید E_i رخداد آمدین اولین 4 شیر متوالی با شروع از پرتاب i ام باشد. داریم

$$Pr[E] = Pr[E_1] + Pr[E_2] + Pr[E_3] + Pr[E_4] + Pr[E_5] + Pr[E_6] + Pr[E_7]$$

از این میان $Pr[E_1]$ و $Pr[E_3]$ را محاسبه می‌کنیم. بقیه مشابه این قابل محاسبه است.

$$Pr[E_1] = (\frac{1}{2})^6, \quad Pr[E_3] = (\frac{1}{2})^5$$

۵. دو نفر پشت سر هم بازی می‌کنند. احتمال برد در هر رقابت $\frac{1}{2}$ است و مستقل از بازیهای قبلی است. هر فردی که تعداد بردش به n رسید بازی تمام می‌شود. با چه احتمالی فرد بازنده در k رقابت برنده شده است؟

فرض کنید E رخداد مورد نظر باشد. با توجه به توصیف مسئله، در اتمام کار، تعداد بازیها باید $n + k$ باشد. نفر اول را با H و نفر دوم را با T نشان می‌دهیم. با فرض اینکه H برنده باشد، آخرین بازی را حتما H برده است. پس

$$\underbrace{\cdots\cdots\cdots\cdots}_{k \text{ number of T's, } n-1 \text{ number of H's}} H$$

$$Pr[E \mid \text{H wins }] = \binom{n+k-1}{k}(\frac{1}{2})^{k+n-1}$$

$$Pr[E] = Pr[\text{ H wins }]Pr[E \mid \text{ H wins }] + Pr[\text{ T wins }]Pr[E \mid \text{ T wins }] = \binom{n+k-1}{k}(\frac{1}{2})^{k+n-1}$$

۶. فرض کنید که یک سکه اریب biased داریم که می‌دانیم احتمال آمدن شیر p است. یعنی

$$Pr[\text{خط}] = 1 - p, \qquad Pr[\text{شیر}] = p$$

نشان دهید چگونه می‌توان با استفاده از این سکه اریب، پرتاب سکه نااریب unbiased را شبیه‌سازی کرد. برای سکه نااریب داریم

$$Pr[\text{خط}] = \frac{1}{2} = Pr[\text{شیر}]$$

(راهنمایی: سکه اریب را چند بار پرتاب کنید.) بطور متوسط سکه اریب را چندبار باید پرتاب کنیم تا یک پرتاب سکه نااریب شبیه‌سازی شود؟ در دو پرتاب سکه اریب، احتمال پرتاب اول شیر باشد و پرتاب دوم خط باشد برابر با $p(p-1)$ است. به همین ترتیب، احتمال پرتاب اول خط و پرتاب دوم شیر باشد برابر با $(p-1)p$ است. پس این دو رویداد احتمال یکسان دارند و لذا می‌توانند شیر و خط را شبیه‌سازی کنند. اگر یکی از این رویدادها رخ نداد (نتیجه هر دو پرتاب یکسان باشد) می‌توانیم آزمون را تکرار کنیم. پس با احتمال $2p(p-1)$ آزمون موفق است. متوسط تعداد پرتابها تا اینکه یک آزمون موفق داشته باشیم برابر با $\frac{1}{p(p-1)}$ است.

۷. فرض کنید می‌خواهید عددی تصادفی با توزیع یکنواخت از میان اعداد $\{0, 1, \ldots, n-1\}$ تولید کنید. برای این کار فقط یک سکه سالم در اختیار دارید. برای تولید یک عدد تصادفی از مجموعه مذکور چگونه از سکه استفاده می‌کنید؟ بطور متوسط چند بار سکه را پرتاب می‌کنید؟

یک عدد صحیح در بازه $[0, n-1]$ را می‌توان با $\lceil \log n \rceil$ بیت مشخص کرد. برای انتخاب یک عدد صحیح تصادفی با توزیع یکنواخت از بازه $[0, n-1]$ می‌توانیم به تعداد $\lceil \log n \rceil$ سکه پرتاب کنیم (هر پرتاب سکه یک بیت تصادفی تولید می‌کند.) اگر عدد حاصل کمتر از n باشد، عدد تصادفی مورد نظر را تولید کرده‌ایم در غیر اینصورت آزمون را دوباره تکرار می‌کنیم. به احتمال حداقل $\frac{1}{2}$ آزمون موفق است. لذا متوسط تکرارهای لازم حداکثر 2 می‌باشد. لذا بطور متوسط حداکثر $2\lceil \log n \rceil$ پرتاب سکه انجام داده‌ایم.

۸. در گراف غیر جهتدار G ممکن است چندین برش کمینه سراسری global min-cut وجود داشته باشد. با استفاده از تحلیل الگوریتم Karger (که در کلاس ارائه شد) نتیجه بگیرید که حداکثر $\frac{n(n-1)}{2}$ برش کمینه متفاوت می‌تواند وجود داشته باشد.

در تحلیل مورد نظر که از الگوریتم کارگر ارائه کردیم، یک برش کمینه مشخص را در نظر گرفتیم و ثابت کردیم احتمال اینکه الگوریتم این برش کمینه را بدست آورد حداقل $\frac{1}{n(n-1)}$ است. فرض کنید تعداد برشهای کمینه t باشد. الگوریتم هر کدام را با احتمال حداقل $\frac{1}{n(n-1)}$ بدست می‌آورد. پس تعداد برشهای کمینه نمی‌تواند بیشتر از $n(n-1)$ است.

۹. فرض کنید آیلین و بابک هر دو یک بردار n بعدی (متشکل از اعداد صفر و یک) دارند. فرض کنید A بردار آیلین و B بردار بابک باشد. آیلین و بابک می‌خواهند بدانند که برداری که در اختیار دارند یکسان است یا خیر.

$$A = B \ ?$$

برای این کار، آیلین می‌تواند بردار خود را تمام و کامل به بابک بفرستد و بابک عمل چک را انجام دهد و نتیجه را اطلاع دهد اما انتقال بیتها هزینه‌بر است. راهی پیشنهاد دهید که آیلین و بابک با ارسال تعداد بیت کمتر از n، با احتمال حداقل $1 - 1/k$ تساوی $A = B$ را چک کنند. فرض کنید آیلین و بابک یک منبع بیت تصادفی مشترک دارند (فرض کنید اگر آیلین و بابک به آسمان نگاه کنند هر دو یک رشته بیت کاملا تصادفی یکسان را می‌بینند.)

بابک و آیلین کافی است یک بردار تصادفی صفر و یک Z با طول n را انتخاب کنند (دقت کنید چون منبع تصادفی مشترک دارند هر دو Z را دارند.)

$$A = a_1, \cdots, a_n$$

$$B = b_1, \cdots, b_n$$

$$Z = z_1, \cdots, z_n$$

حال آیلین ضرب داخلی ورودی خود با بردار Z را محاسبه می‌کنند.

$$az = \sum_{i=1}^{n} a_i z_i \ (mod\ 2)$$

بابک هم همین کار را برای ورودی خود انجام می‌دهد.

$$bz = \sum_{i=1}^{n} b_i z_i \ (mod\ 2)$$

دقت کنید اینجا az و bz یک بیت هستند (صفر یا یک). اگر A و B یکسان باشند، روشن است که همواره $az = bz$. از طرف دیگر A و B برابر نباشند می‌توان نشان داد که

$$Pr(az \neq bz \mid A \neq B) = \frac{1}{2}$$

پس بابک و آیلین چند بار این آزمون را بطور مستقل تکرار می‌کنند و az و bz را به هم می‌فرستند. اگر t بار این کار را انجام دهند و $A \neq B$ احتمال اینکه در هر t بار $az = bz$ اتفاق بیافتد برابر با $\frac{1}{2^t}$ است. پس برای اینکه احتمال خطا حداکثر $\frac{1}{k}$ باشد کافی است که بابک و آیلین این آزمون را $\log k$ بار تکرار کنند. هر بار یک بیت را به هم ارسال می‌کنند. در کل $O(\log k)$ بیت ارسال می‌شود.

۱۰. در این تمرین یک الگوریتم تصادفی برای مسئله ۳-SAT پیشنهاد می‌کنیم. می‌خواهیم الگوریتم پیشنهادی را تحلیل کنیم.

توصیف الگوریتم. فرض کنید فرمول ورودی ϕ باشد. با یک مقداردهی تصادفی برای متغیرهای ϕ شروع کن (هر متغیر با احتمال یکسان true یا false است). حال هر بار یک جمله (کلاز) C_i که ارضا نشده است را انتخاب کن و مقدار یکی از متغیرهای C_i را عوض کن. این کار را حداکثر n بار انجام بده. اگر در نهایت همه جمله ها ارضا شدند، گزارش کن که ϕ صدق پذیر است در غیر این صورت گزارش کن که ϕ صدق پذیر نیست.

- با فرض اینکه ϕ صدق پذیر است نشان دهید احتمال اینکه الگوریتم یک مقداردهی تصدیق کننده را پیدا کند حداقل $(\frac{1}{2})(\frac{1}{3})^{n/2}$ است. راهنمایی: فرض کنید A یک مقداردهی باشد که همه جملات فرمول را ارضا کند. با احتمال $\frac{1}{2}$ مقداردهی شروع نصف مقادیر A را درست حدس زده است. در هر قدم با چه احتمالی یکی دیگر از مقادیر درست حدس زده می‌شود؟

فرض کنید H رخدادی باشد که الگوریتم در شروع کار نصف مقادیر A را درست حدس زده است. داریم

$$Pr(H) = \frac{1}{2}$$

اگر الگوریتم خوش شانس باشد، در قدم اول یک متغیر که حدسش درست نبوده مقدارش عوض می‌شود. چون عبارت از نوع ۳-SAT است شانس انتخاب متغیر مورد نظر حداقل $\frac{1}{3}$ است. لذا بعد از قدم اول، احتمال اینکه $\frac{n}{2}+1$ متغیر درست حدس زده شوند حداقل $(\frac{1}{2})(\frac{1}{3})$ است. بعد از قدم دوم، احتمال اینکه $\frac{n}{2}+2$ متغیر درست حدس زده شوند، برابر با $(\frac{1}{2})(\frac{1}{3})^2$ است. لذا بعد از $\frac{n}{2}$ قدم احتمال اینکه n متغیر (همه متغیرها) درست حدس زده شوند $(\frac{1}{2})(\frac{1}{3})^{n/2}$ است.

- با توجه به گزاره قبلی، با فرض اینکه فرمول ورودی صدق پذیر باشد، الگوریتم بالا را چقدر باید تکرار کنیم تا به احتمال حداقل $\frac{3}{4}$ جواب درست بدهیم؟

فرض کنید احتمال موفقیت در یک تکرار مستقل p باشد. در این مثال $p = (\frac{1}{2})(\frac{1}{3})^{n/2}$. اگر F رخداد این باشد که در t تکرار مستقل هیچ موفقیتی بدست نیاید.

$$Pr(F) = (1-p)^t$$

لذا S رخداد این باشد که در t تکرار حداقل یک موفقیت بدست آید، داریم

$$Pr(S) = 1-(1-p)^t$$

لذا اینجا داریم

$$Pr(F) = (1-(\frac{1}{2})(\frac{1}{3})^{n/2})^t \leq \frac{1}{4}$$

پس باید t را جوری انتخاب کنیم که نتیجه بالا بدست بیاید. برای ساده شدن از نامساوی زیر استفاده می‌کنیم.

$$1-p \leq e^{-p}$$

لذا

$$(1-(\frac{1}{2})(\frac{1}{3})^{n/2})^t \leq (e^{-(\frac{1}{2})(\frac{1}{3})^{n/2}})^t \leq \frac{1}{4}$$

بدست می‌آید

$$t \geq \frac{\ln 4}{(\frac{1}{2})(\frac{1}{3})^{n/2}} = O(1.733^n)$$

- الگوریتم پیشنهادی از چه نوعی است؟ RP ، CoRP ، ZPP و شاید هیچ کدام.
  هیچکدام چون زمان چند جمله‌ای ندارد.