طراحی الگوریتم - پاسخ‌نامه تکلیف چهارم

پیش از حل سوالات به موارد زیر دقت کنید:

- الگوریتم‌های برنامه‌ریزی پویای متفاوتی که خروجی صحیح داشته باشند، نمره سوال را دریافت می‌کنند.
- در صورت وجود هرگونه ابهام می‌توانید در گروه تلگرام یا گروه اسکایپ سوالات خود را مطرح کنید.
- از طریق ایمیل‌های زیر می‌توانید با TAهای مربوط به این تکلیف در ارتباط باشید.
  - سوال ۱ تا ۳: mnaeimi+algo@ec.iut.ac.ir
  - سوال ۴ تا ۶: kazemimaryam1998@gmail.com
- تصحیح و نمره‌دهی این تکلیف به زودی انجام می‌شود.

سوال ۱. (۱۵ نمره) m نوع اسکناس $(d_i \in \mathbf{N})$ به صورت $d_1 < d_2 < \cdots < d_m$ در اختیار داریم. فرض کنید از هر نوع اسکناس به هر تعداد لازم در اختیار داریم و همچنین $d_1 = 1$. الگوریتم برنامه ریزی پویا ارائه دهید که با دریافت آرایه‌ای از انواع اسکناس ها و مقدار n، کمترین تعداد اسکناس لازم برای پرداخت این مقدار (n) را به دست آورد. پیچیدگی زمانی الگوریتم خود را بررسی نمایید.

- $F(n)$ را کمترین تعداد اسکناس که مجموع مقدارشان برابر n می‌شود در نظر بگیرید.
- مقدار n فقط می‌تواند با اضافه‌کردن یک اسکناس با مقدار d_j به مقدار $n - d_j$ به ازای $j = 1, 2, ..., m$ که $n \geq d_j$.
- می‌توانیم هر حالت ممکن را بررسی کنیم و حالتی که مقدار $F(n - d_j) + 1$ را کمینه می‌کند را انتخاب نماییم.
- برای به‌دست آوردن $F(n)$ از آنجایی که 1 مقدار ثابتی است می‌توانیم مقدار کمینه را برای $F(n - d_j)$ به‌دست آوریم و آن را با مقدار 1 جمع کنیم.
- رابطه بازگشتی به صورت زیر به دست می‌آید:

$$F(n) = \min_{j:n \geq d_j}\{F(n - d_j)\} + 1, \quad n > 0$$
$$F(0) = 0$$

- پیچیدگی زمانی این الگوریتم با توجه به اینکه برای هر مرحله، حداکثر تمامی اسکناس‌های مختلف را مورد بررسی قرار می‌دهد برابر است با $O(nm)$.

سوال ۲. (۲۰ نمره) در بعضی از خانه‌های یک صفحه بازیِ $n \times m$، یک سکه قرارگرفته‌است. یک نمونه صفحه از این بازی به صورت آرایه $n \times m$ در اختیار شما قرار می‌گیرد، مقدار آن در خانه‌های دارای سکه 1 و در خانه‌های خالی 0 می‌باشد. مهره شما در خانه بالا چپ $(1, 1)$ صفحه قرار دارد و تنها می توانید به سمت راست یا به سمت پایین حرکت نمایید تا به خانه پایین راست (n, m) برسید. یک الگوریتم برنامه‌ریزی پویا بنویسید که با دریافت یک نمونه آرایه از صفحه، حداکثر تعداد سکه‌ای که ممکن‌است در این نمونه بازی جمع‌آوری شود را محاسبه نماید.

- B را آرایه صفحه در نظر بگیرید.
- $F(i, j)$ را بیشترین تعداد سکه‌ای که می‌توانید با رساندن مهره به خانه (i, j) در ردیف i و ستون j صفحه، به‌دست آورید در نظر بگیرید.
- مهره شما برای اینکه در هر خانه (i, j) قرار بگیرد باید از خانه مجاور بالایی $(i - 1, j)$ یا از خانه مجاور سمت چپ $(i, j - 1)$ وارد آن شود.
- از آنجا که خانه‌های ردیف اول خانه مجاور بالایی ندارند و خانه‌های ستون اول خانه مجاور سمت چپ ندارند این مقدار را برای آنها برابر صفر در نظر می‌گیریم.
- رابطه بازگشتی به صورت زیر به‌دست می‌آید:

$$F(i, j) = \max\{F(i - 1, j), F(i, j - 1)\} + B_{ij}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m$$
$$F(0, j) = 0, \quad 1 \leq j \leq m \qquad\qquad F(i, 0) = 0, \quad 1 \leq i \leq n$$

سوال ۳. (۲۵ نمره) قصد جابه‌جایی یک محموله حساس را داریم که هربار باید حداکثر تا قبل از فاصله معینی (D) مورد بازرسی قرار بگیرد. باید زیرمجموعه‌ای از گره‌ها را برای بازرسی در طول مسیر انتخاب نماییم. یک مسیر $< v_1, v_2, v_3, \cdots, v_n >$ به ما داده شده است که شامل n ایستگاه بازرسی است. هر گره (Node) یک ایستگاه بازرسی را نمایش می‌دهد. برای هر گره v_i $(1 \leq i \leq n)$، یک هزینه بازرسی $c(v_i)$ معین شده‌است. همچنین می‌دانیم $c(v_1) = c(v_n) = 0$. به علاوه برای هر یال (v_i, v_{i+1}) که $(1 \leq i \leq n-1)$، فاصله $d(v_i, v_{i+1})$ داده شده‌است که از مقدار معین D کمتر است. ($d(v_i, v_{i+1}) \leq D$: $(\forall(1 \leq i \leq n-1)$

در هر مسیر، فاصله بین هر دو جفت گره v_i و v_j $(i < j)$ برابر است با مجموع فاصله یال‌ها در مسیر بین دو گره. یک زیرمجموعه انتخابی از گره‌ها به صورت $V' \subseteq \{v_1, v_2, \cdots, v_n\}$، «قابل قبول» خواهدبود اگر فاصله بین هر دو جفت گره متوالی در $V' \cup \{v_1, v_n\}$ حداکثر برابر با D باشد. هزینه هر مسیر برابر است با مجموع هزینه بازرسی‌های گره‌های داخل V'. یک زیرمجموعه V^* از گره‌ها انتخابی بهینه است، اگر قابل قبول باشد و کمترین هزینه را در میان همه انتخاب‌های قابل قبول داشته‌باشد.

یک الگوریتم برنامه‌ریزی پویا ارائه دهید که هزینه انتخاب بهینه را محاسبه نماید. ارائه خود مجموعه گره‌های انتخابی لازم نیست. همچنین نیاز به اجرای الگوریتم خود را روی یک مثال نیست. پیچیدگی زمانی الگوریتم خود را نیز بررسی کنید.

- هر قسمت از مسیر به شکل $< v_1, v_2, ..., v_i >$، $1 \leq i \leq n$ یک زیرمسئله را تشکیل می‌دهد.
- $P(i)$ را کمترین هزینه برای زیرمجموعه قابل قبول برای زیرمسئله iام، که شامل نودهای v_1 و v_i می‌باشد، در نظر بگیرید.
- از آنجا که $c(v_1) = 0$ پس $P(1) = 0$.
- برای هر زیرمسئله دیگر چون $i \geq 2$ است، همیشه حداقل یک ایستگاه بازرسی قبل از v_i وجود دارد.
- در هر انتخاب بهینه برای قسمت iام، ایستگاه بازرسی یکی مانده به آخر (ایستگاه قبل از v_i) انتخاب شده باید حداکثر در فاصله D قرار داشته باشد.
- پس برای نود یکی مانده به آخر، تنها نودهایی را در نظر می‌گیریم که فاصله آنها از نود v_i کمتر یا مساوی با D باشد.
- رابطه بازگشتی به صورت زیر به‌دست می‌آید:

$$P(i) = c(v_i) + \min_{1 \leq j \leq i-1}\{P(v_j) : d(v_j, v_i) \leq D\}, \quad i \geq 2$$
$$P(1) = 0$$

- در صورتی که در الگوریتم فوق برای محاسبه فاصله ها یکبار فاصله تمامی نودها از نود اول را در یک آرایه ذخیره کنیم،هربار می‌توانیم فاصله دو نود را در زمان $O(1)$ به‌دست آوریم.
- برای محاسبه هر $P[i]$ باید کمترین مقدار از میان تمامی $i-1$ حالت را محاسبه نماییم. پس پیچیدگی زمانی این الگوریتم برابر $O(n^2)$ می‌باشد.

سوال ۴. (۲۰ نمره) درخت کوتاه‌ترین مسیر از هر گره به گره‌ی ۱ برای گراف زیر را با استفاده از الگوریتم Bellman-Ford به دست آورید. جدول زیر را کامل کنید.

- فرض کنید $C_l{}^h$ برابر با هزینه از نود l به نود 1 در تکرار hام باشد.

- $\forall h : C_1{}^h = 0$
- $\forall i \neq 1 : C_i{}^1 = d_{1i}$
- $\forall i \neq 1 : C_i{}^{h+1} = \min_j\{C_j{}^h + d_{ji}\}$

- یال های درخت کوتاه‌ترین مسیر بعد از اجرای الگوریتم بلمن-فورد محاسبه می‌شوند.
- برای هر $i \neq 1$ یالی را به درخت کوتاه‌ترین مسیر اضافه می‌کنیم که معادله بلمن را کمینه می‌کند.

i	C_i^1	C_i^2	C_i^3	C_i^4	C_i^5	Shortest path arcs†
1	0	0	0	0	0	
2	4	4	4	4	4	(1, 2)
3	5	5	5	5	5	(1, 3)
4	∞	7	7	7	7	(2, 4)
5	∞	14	13	12	12	(6, 5)
6	∞	14	10	10	10	(1, 2)
7	∞	∞	16	12	12	(1, 2)

سوال ۵. (۲۰ نمره) سینا و عماد تصمیم دارند که به صورت قاچاقی به سفر بروند. در این نوع سفر، n قاچاقچی وجود دارد که با شماره های ۱ تا n مشخص شده‌اند. قاچاقچی ۱ و قاچاقچی n، قاچاقچی‌های مبدا و مقصد هستند و هدف سینا و عماد، جابه‌جایی از قاچاقچی مبدا به قاچاقچی مقصد است. هزینه انتقال از قاچاقچی i به قاچاقچی j که در آن $1 \leq i < j \leq n$ را با c_{ij} نشان می‌دهیم. دقت کنید که هر قاچاقچی تنها می‌تواند عماد و سینا را به یک قاچاقچی با شماره بزرگتر برساند. همچنین c_{ij} ها قاعده‌ی خاصی ندارند، مثلا ممکن است که $c_{24} = 10$ و $c_{26} = 3$ باشد. یک الگوریتم برنامه‌ریزی پویا ارائه دهید تا کمترین هزینه‌ای را که عماد و سینا باید بپردازند را محاسبه کند. همچنین پیچیدگی زمانی این الگوریتم را بررسی کنید.

- اگر $m[i]$ را کمینه هزینه انتقال از قاچاقچی i به قاچاقچی n نظر بگیریم، جواب سوال برابر با $m[1]$ خواهد بود.
- پس برای $i = n$ داریم $m[i] = 0$.
- رابطه بازگشتی به صورت زیر به دست می‌آید:

$$m(i) = \min_{i<j<n}\{c_{ij} + m[j]\}, \quad i \neq n$$

- در این صورت کمترین هزینه ممکن را برای جایگاه‌های $n-1$، $n-2$ و... حساب می‌کنیم تا در نهایت به $m[1]$ برسیم و در هنگام محاسبه‌ی $m[1]$، تمامی mها از پیش حساب شده‌اند. همچنین با این روند برای محاسبه‌ی $m[i]$، نیاز است که کمینه $n-i$ عدد را محاسبه کنیم. پس در مجموع $\sum_{i=1}^{n-1} i$ عمل انجام می‌دهیم که از $O(n^2)$ است.

سوال ۶. (۲۵ نمره - اختیاری) کشوری فرضی n شهر دارد که میان هر دو شهر آن، دو جاده وجود دارد: جاده اول از شهر اول به شهر دوم و جاده دوم از شهر دوم به شهر اول. مدت زمانی که طول می کشد در مسیر شهر i به شهر j حرکت کنیم را با t_{ij} نشان می‌دهیم. دقت کنید که لزومی ندارد که t_{ij} با t_{ji} برابر باشد. همچنین کوتاه‌ترین مسیر بین دو شهر i و j لزوما مسیر مستقیم ij نیست. این کشور درگیر یک جنگ می شود و طی آن تعدادی از شهرها به دست دشمن می افتد. در نتیجه راه ارتباطی شهرها از طریق شهرهایی که به دست دشمن می‌افتد، بسته می شود و ممکن است کوتاه‌ترین فاصله‌ی میان دو شهر نیز تغییر کند. مسئولین این کشور تمایل دارند بدانند پس از تصرف هر شهر، کوتاه‌ترین مسیر از شهر i به j (که تصرف نشده‌اند) چه مقدار خواهد بود. یک الگوریتم برنامه‌ریزی پویا از مرتبه زمانی $O(n^3)$ ارائه دهید که با دریافت شهرهای تصرف‌شده و زمان‌بندی‌ها بتواند حداقل مدت زمان حرکت بین هر دو شهر دلخواه را محاسبه نماید.

- در یک آرایه شهرهایی که تصرف شده‌اند را ذخیره می‌کنیم.
- با استفاده از فواصل اولیه شهرها، الگوریتم فلوید-وارشال را بر روی شهرهای باقی‌مانده اجرا می‌کنیم.
- شهرهای تصرف شده در حین این عمل می‌توانند به عنوان مبدا و مقصد در نظر گرفته شوند، فقط نمی‌توانند شهر واسط در الگوریتم فلوید-وارشال باشند.
- با توجه به پیچیدگی زمانی الگوریتم فلوید-وارشال، پیچیدگی زمانی روش فوق نیز $O(n^3)$ می‌باشد.