طراحی و تحلیل الگوریتم

استاد:

دکتر زاهد رحمتی

تدریسیاران:

اشکان ودادی

داریوش کاظمی

ترم دوم ۱۴۰۰

جلسه چهارم

برنامه نویسی پویا (DP)

خلاصه مطلب امروز:

- توضیح برنامه نویسی پویا
- راهنمای پیاده سازی برنامه نویس پویا
- مسئله 1 – ضریب دو جمله‌ای
- مسئله 2 – کوله پشتی 0 و 1
- مسئله 3 – نصب پوستر

روش DP:

در روش پویا ابتدا نمونه‌های کوچکتر را حل کرده و نتایج را **ذخیره** می‌کنیم. بعدا هر زمانی به آن‌ها نیاز شد به جای دوباره حساب کردن، از داده ذخیره شده استفاده می‌کنیم.

- در DP از جدول یا آرایه استفاده میشود.
- مشابه تقسیم و حل

1- پیدا کردن رابطه بازگشتی
2- حل نمونه از مسئله به شیوه پایین به بالا با حل نمونه‌های کوچکتر (یا از پایین به بالا)

هر مسئله بهینه‌سازی را نمی‌توان با استفاده از برنامه نویسی پویا حل کرد. اصل بهینگی باید در مسئله صدق کند. گفته می‌شود اصل بهینگی در یک مسئله صدق می‌کند اگر یک حل بهینه برای نمونه‌ای مسئله، همواره حاوی حل بهینه برای همه زیرنمونه‌ها باشد.

روش پیاده سازی DP:

مرحله اول: پیدا کردن رابطه بازگشتی برای مسئله
مرحله دوم: مشخص کردن ابعاد و اندازه جدول (آرایه، لیست)
مرحله سوم: تعریف هر خانه جدول با کمک رابطه بازگشتی
مرحله چهارم: مقداردهی اولیه و شرایط مرزی
مرحله پنجم: نحوه بروزرسانی خانه‌ها
مرحله ششم: مشخص کردن جواب‌ها

ضریب دو جمله ای:

محاسبه ضریب جمله k+1-ام در بسط $(a+b)^n$:

$$0 \leq k \leq n$$

$$\binom{n}{k} = \frac{(n!)}{k!\,(n-k)!}$$

ضریب دو جمله ای:

محاسبه ضریب جمله k+1-ام در بسط $(a+b)^n$:

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k}, & 0 < k < n \\ 1, & k = 0 \; or \; k = n \end{cases}$$

ضریب دو جمله ای:

محاسبه ضریب جمله k+1-ام در بسط $(a+b)^n$:

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k}, & 0 < k < n \\ 1, & k = 0 \ or \ k = n \end{cases}$$

ویژگی بازگشتی میتوان استفاده کرد و ضابطه زیر را نوشت:

$$B[i][j] = \begin{cases} B[i-1][j-1] + B[i-1][j], & 0 < j < i \\ 1, & j = 0 \ or \ i = n \end{cases}$$

ضریب دو جمله ای:

$$B[i][j] = \begin{cases} B[i-1][j-1] + B[i-1][j], & 0 < j < i \\ 1, & j = 0 \ or \ i = n \end{cases}$$

به عنوان مثال میخواهیم $\binom{4}{2}$را حل کنیم. معادل بدست آوردن B[4][2] است.

ضریب دو جمله ای:

$$B[i][j] = \begin{cases} B[i-1][j-1] + B[i-1][j], & 0 < j < i \\ 1, & j = 0 \ or \ i = n \end{cases}$$

به عنوان مثال میخواهیم $\binom{4}{2}$را حل کنیم. معادل بدست آوردن B[4][2] است.
ابتدا یک ماتریس 3*4 می‌سازیم.

1		
1	1	
1		1
1		
1		

ضریب دو جمله ای:

$$B[i][j] = \begin{cases} B[i-1][j-1] + B[i-1][j], & 0 < j < i \\ 1, & j = 0 \ or \ i = n \end{cases}$$

به عنوان مثال میخواهیم $\binom{4}{2}$ را حل کنیم. معادل بدست آوردن B[4][2] است.
ابتدا یک ماتریس (2+1)*4 می‌سازیم.

$$B[2][1] = B[1][0] + B[1][1] = 1 + 1 = 2$$

$$B[3][1] = B[2][0] + B[2][1] = 1 + 2 = 3$$

$$B[3][2] = B[2][1] + B[2][2] = 2 + 1 = 3$$

$$B[4][1] = B[3][0] + B[3][1] = 1 + 3 = 4$$

$$B[4][2] = B[3][1] + B[3][2] = 3 + 3 = 6$$

1		
1	1	
1	2	1
1	3	3
1	4	6

ضریب دو جمله ای:

$$B[i][j] = \begin{cases} B[i-1][j-1] + B[i-1][j], & 0 < j < i \\ 1, & j = 0 \ or \ i = n \end{cases}$$

پیچیدگی زمانی:

$$1 + 2 + 3 + \cdots + k + \overbrace{(\boldsymbol{k+1}) + (\boldsymbol{k+1}) + \cdots + (\boldsymbol{k+1})}^{\text{بار}\, n-k+1}$$

$$\frac{k(k+1)}{2} + (n-k+1)(k+1) = \frac{(2n-k+2)(k+1)}{2} \in \theta(nk)$$

کوله‌پشتی 0-1

دزدی وارد یک جواهر فروش شده و می‌خواهد قطعه‌هایی که دارای ارزش و وزن معینی هستند را طوری در کوله پشتی خود قرار دهد که **بیشترین سود** حاصل شود

- البته وزن قطعه‌ها از یک حد مشخص نباید بیشتر شود، چون کوله پشتی پاره خواهد شد.
- اگر قطعه‌ها به گونه‌ای باشد یا انتخاب می‌شوند یا نه میگوییم کوله پشتی 0 و 1!

- کوله پشتی صفر و یک: DP
- کوله پشتی کسری: حریصانه

کوله‌پشتی 0–1

دزدی وارد یک جواهر فروش شده و می‌خواهد قطعه‌هایی که دارای ارزش و وزن معینی هستند را طوری در کوله پشتی خود قرار دهد که **بیشترین سود** حاصل شود

- البته وزن قطعه‌ها از یک حد مشخص نباید بیشتر شود، چون کوله پشتی پاره خواهد شد.
- اگر قطعه‌ها به گونه‌ای باشد یا انتخاب می‌شوند یا نه میگوییم کوله پشتی 0 و 1!

- DP

در هر مرحله بررسی می‌کند که دو حالت ممکن را:

1. اگر جنس را بردارد $P_n + knapsack(W - w_n, n - 1)$
2. اگر جنس را برندارد $knapsack(W, n - 1)$

کوله‌پشتی 0–1

دزدی وارد یک جواهر فروش شده و می‌خواهد قطعه‌هایی که دارای ارزش و وزن معینی هستند را طور ی در کوله پشتی خود قرار دهد که **بیشترین سود** حاصل شود

الگوریتم:

1. ایتم‌ها را مرتب کنیم به صورت صعودی
2. ماتریس P[n][w] با اندازه (تعداد موارد+1)*(ظرفیت کوله پشتی+1)
3. P_n: پول هر آیتم W_n: وزن آیتم‌ها

کوله‌پشتی 0-1

دزدی وارد یک جواهر فروش شده و می‌خواهد قطعه‌هایی که دارای ارزش و وزن معینی هستند را طور ی در کوله پشتی خود قرار دهد که **بیشترین سود** حاصل شود

الگوریتم:

1. ایتم‌ها را مرتب کنیم به صورت صعودی
2. ماتریس P[n][w] با اندازه (تعداد موارد+1)*(ظرفیت کوله پشتی+1)
3. P_n: پول هر آیتم w_n: وزن آیتم‌ها

(اگر جنس را بردارد$(1 - knapsack(W - w_n, n - 1$ + P_n یا اگر جنس را برندارد $(1 - knapsack(W, n$)

$$\boldsymbol{P[n][w]} = \begin{cases} \mathbf{max}(\boldsymbol{P[n-1][w]}, \boldsymbol{P_n} + \boldsymbol{P[n-1][w-w_n]}, w_n \le W \\ \boldsymbol{P[n-1][W]}, w_n > W \end{cases}$$

توجه: تنها عناصر مورد نیاز در سطر (n-1)-ام، آنهایی هستند که برای محاسبه$P[n][w]$ به کار می‌روند. که عبارتند از: $P[n-1][w]$ و $P[n-1][w-w_n]$

کوله‌پشتی 0–1

دزدی وارد یک جواهر فروش شده و می‌خواهد قطعه‌هایی که دارای ارزش و وزن معینی هستند را طور ی در کوله پشتی خود قرار دهد که **بیشترین سود** حاصل شود

$$\boldsymbol{P[n][w]} = \begin{cases} \mathbf{max}(\boldsymbol{P[n-1][w], P_n + P[n-1][w-w_n]}), w_n \leq W \\ \boldsymbol{P[n-1][W]}, w_n > W \end{cases}$$

توجه: تنها عناصر مورد نیاز در سطر (n–1)–ام، آنهایی هستند که برای محاسبه $P[n][w]$ به کار می‌روند. که عبارتند از:
$P[n-1][w]$ و $P[n-1][w-w_n]$

پیچیدگی زمانی:

$$\boldsymbol{O\big(minimum(2^n, nW)\big)}$$

فرادرس: تا کنون الگوریتمی برای بدترین حالت بهتر از روش DPارائه نشده است و در عین حال عدم امکان آن را نیز کسی ثابت نکرده است.

کوله‌پشتی 0–1

$$\boldsymbol{P}[\boldsymbol{n}][\boldsymbol{w}] = \begin{cases} \mathbf{max}(\boldsymbol{P}[\boldsymbol{n}-\mathbf{1}][\boldsymbol{w}], \boldsymbol{P}_{\boldsymbol{n}} + \boldsymbol{P}[\boldsymbol{n}-\mathbf{1}][\boldsymbol{w}-\boldsymbol{w}_{\boldsymbol{n}}], w_n \leq W \\ \boldsymbol{P}[\boldsymbol{n}-\mathbf{1}][\boldsymbol{W}], w_n > W \end{cases}$$

مثال) با فرض w = 30، سود بهینه را بدست آورید. طبق الگوریتم باید جواب P[3][30] را بدست آوریم.

وزن (پوند)	ارزش (دلار)	قطعه
5	50	1
10	60	2
20	140	3

کوله‌پشتی 0–1

قطعه	ارزش (دلار)	وزن (پوند)
1	50	5
2	60	10
3	140	20

مثال) با فرض w = 30، سود بهینه را بدست آورید. طبق الگوریتم باید جواب P[3][30] را بدست آوریم.

$$P[n][w] = \max(P[n-1][w], P_n + P[n-1][w-w_n])$$

$$P[3][30] = \text{maximum}\begin{cases} P[2][30] \\ P_3 + P[2][30-w_3] = 140 + P[2][10] \end{cases}$$

$$P[2][10] = \text{maximum}\begin{cases} P[1][10] = 50 \\ P_2 + P[1][10-w_2] = 60 + P[1][0] = 60 \end{cases} \Rightarrow P[2][10] = 60$$

$$P[2][30] = \text{maximum}\begin{cases} P[1][30] = 50 \\ P_2 + P[1][30-w_2] = 60 + P[1][20] = 110 \end{cases} \Rightarrow P[2][30] = 110$$

$$p[3][30] = \text{maximum}\begin{cases} P[2][30] = 110 \\ 140 + P[2][10] = 140 + 60 = 200 \end{cases} \Rightarrow p[3][30] = 200$$

مسئله نصب پوستر:

در یک راهرو n تابلو پشت سر هم برای نصب پوستر آماده شده است. (تابلوهای b1 تا bn)
طبق قوانین، یک پوستر نباید روی دو تا تابلو پشت سر هم نصب شود و در یک تابلو نباید بیش از یک پوستر نصب شود.

برای هر تابلو یک اهمیت دید (Wi) تعیین شده که نشان دهنده‌ی میزان دید آن تابلو است. (هر مقدار این عدد بزرگتر باشد، به این معنی است که پوستر این تابلو بیشتر از بقیه دیده می‌شود)

با داشتن Wi ها بری تمام تابلوها می‌خواهم یک پوستر را در تعدادی از این تابلو ها نصب کنیم که مجموع اهمیت دید آن بهینه شود.

W1 ~~W2~~ **W3**
~~W1~~ **W2** ~~W3~~

مسئله نصب پوستر:

برای حل این مسئله، هر پوستری که در نظر بگیرید دو حالت می‌تواند رخ دهد:
الف– این پوستر روی تابلو i نصب بشود.

ب– این پوستر روی تابلو i نصب نشود.

مسئله نصب پوستر:

برای حل این مسئله، هر پوستری که در نظر بگیرید دو حالت می‌تواند رخ دهد:

الف– این پوستر روی تابلو i نصب بشود:

دیگر نمی‌تواند روی تابلوی i-1 ام نصب شود پس مجموع اهمیت دیدها: Wi+f(i-2)

ب– این پوستر روی تابلو i نصب نشود.

مجموع اهمیت دیدها: f(i-1)

مسئله نصب پوستر:

برای حل این مسئله، هر پوستری که در نظر بگیرید دو حالت می‌تواند رخ دهد:

الف– این پوستر روی تابلو i نصب بشود:

دیگر نمی‌تواند روی تابلوی i-1 ام نصب شود پس مجموع اهمیت دیدها: Wi+f(i-2)

ب– این پوستر روی تابلو i نصب نشود.

مجموع اهمیت دیدها: f(i-1)

$$f(i) = \max \{ f(i-1), W_i + f(i-2)\}$$

خسته نباشید!

داریوش کاظمی – اشکان ودادی