حدیث غفوری ۹۸۲۰۵۴۱۳

تکلیف ۴ الگوریتم

$\boxed{\text{سوال ۶}}$

m نوع اسکناس $(d_i \in N)$ به صورت $d_1 < d_2 < \dots < d_m$ و

$\boxed{d_1 = 1}$

ورودی: آرایه‌ای از انواع اسکناس‌ها و $n \leftarrow d[m]$ و n

خروجی: min تعداد اسکناس برای پرداخت n

راه حل: یک آرایه به اسم $\boxed{ans}$ تعریف می‌کنیم (با سایز n+1) ← $ans[n+1]$

در $ans[i]$، حداقل تعداد اسکناس قرار می‌گیرد

در حالت کلی ← جواب: $\boxed{ans[n]}$

حالت پایه: برای مقدار صفر ← هیچ اسکناس نیاز نیست. ← $\boxed{ans[0]=0}$

رابطه‌ی کلی:

$$ans[i] = \min\{d_1 + ans[i - d[j]]\}$$

برای $j=0$ تا $j=m$ (برای کل اسکناس‌ها)

← جواب: $ans[n]$

$\boxed{\text{Time complexity} = O(nm)}$

دو حلقه‌های تو در تو با سایزهای m, n → $\boxed{O(nm)}$

```
for i=1 to n
    ans[i] = ∞
ans[0] =0
for i=1 to n
    for j=0 to m
        if( d[j] ≤ i)
            ans[i]= min{ans[i],
                        ans[i-d[j]]+1}
return ans[n]
```

$\boxed{\text{سوال ۲}}$

یک صفحه بازی $\boxed{n \times m}$

دارای سکه ← ۱ $\boxed{1}$

فاقد سکه ← مقدار داخل خانه $\boxed{0}$

شروع از $(1,1)$ ← تنها به راست و پایین می‌توانیم حرکت کنیم

مقصد $\boxed{(n,m)}$

ورودی: یک آرایه از صفحه ← مثلاً $S[n][m]$

خروجی: حداکثر تعداد سکه

راه حل: آرایه‌ی ans (آرایه‌ی دو بعدی) با اندازه‌های $ans[n+1][m+1]$ → بیشترین مقدار سکه‌ای که جمع می‌شود در خانه‌ی $ans[i][j]$ است.

جواب: $ans[n][m]$

پس ابتدا $ans[i][0] = 0$ ← $0 \le i \le n$ سطرها

و $ans[0][j] = 0$ ← $0 \le j \le m$ ستون‌ها

برای رسیدن به خانه‌ی (i,j) ← از خانه‌ی $(i-1, j)$ ← خانه‌ی بالایی می‌رویم

یا از خانه‌ی $(i, j-1)$ — خانه‌ی سمت چپ

چون فقط به سمت راست و پایین می‌توانیم حرکت کنیم

خانه بالایی — خانه چپی

$$ans[i][j] = \max\{ans[i-1][j], ans[i][j-1]\} + S[i][j]$$

مقدار داخل خانه (صفر یا یک)

جواب در $ans[n][m]$ ذخیره می‌شود

Time complexity: $\boxed{O(nm)}$

← مثل مسئله‌ی قبلی ۲ حلقه‌ی تو در تو

سنگ
فاصله

سوال 3

معمولا باید هر بار قبل از $\boxed{D}$ ، بازرسی شود.

یک مسیر به ما داده شده که شامل n ایستگاه - هر گره ← ایستگاه

$\langle v_1, v_2, \ldots, v_n \rangle$

v_i و $1 < i < n$

یک هزینه دارد $c(v_i)$

و $c(v_1) = c(v_n) = 0$

برای هر یال (v_i, v_{i+1})

↓

$d(v_i, v_{i+1})$ داده شده

فاصله

فاصله هر دو گره = مجموع فاصله یال ها در مسیر بین گره ها

زیر مجموعه قابل قبول e اگر فاصله بین هر دو گره متوالی e حداکثر D باشد.

هزینه هر مسیر = مجموع هزینه بازرسی های گره های v'

انتخاب بهینه e اگر قابل قبول باشد و کمترین هزینه را در میان همه انتخاب های قابل قبول داشته باشد.

D و $c(v_i)$, $d(v_i, v_{i+1})$ ورودی e

خروجی e هزینه ی انتخاب بهینه

راه حل e یک آرایه به نام masir[n][n] تعریف می کنیم

masir[i][j] ، طول مسیر از i به j را نشان می دهد.

$masir[i][j] = masir[i][j-1] + d(v_{j-1}, v_j)$

سپس آرایه ی ans[n][n] را تعریف می کنیم که

ans[i][j] = هزینه بازرسی تا $\boxed{j}$ با رفتن از i به j

```
if (masir[i][j] > D)
    ans[i][j] = ∞
else
    ans[i][j] = c_j + min{ans[k][i]}
```

$\boxed{1 \le k \le i-1}$

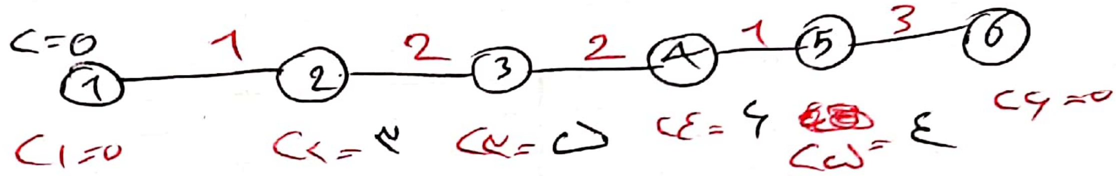
جواب مسئله $\min\{ans[i][n]\}$ که $\boxed{1 < i < n}$

Time complexity: $O(n^3) = O(n^2 \times n)$

n^2 برای پرکردن جدول ، ۲ حلقه‌ای که در کدنویسی داریم

برای پیدا کردن ans[i][j] باید بین ۲ عدد کمترینش را انتخاب کنیم → $\boxed{O(n)}$

مثال: $\boxed{n=۵}$



	1	2	3	4	5	6
1	0	1	3	5	6	9
2	X	∞	2	4	5	8
3	X	X	∞	2	3	6
4	X	X	X	∞	1	4
5	X	X	X	X	∞	3
6	X	X	X	X	X	∞

بیشتر از n ← ∞ می‌شود

	1	2	3	4	5	6
1	∞	3	5	6	∞	∞
2	X	∞	8	9	7	∞
3	X	X	∞	11	9	∞
4	X	X	X	∞	10	6
5	X	X	X	X	∞	7
6	X	X	X	X	X	∞

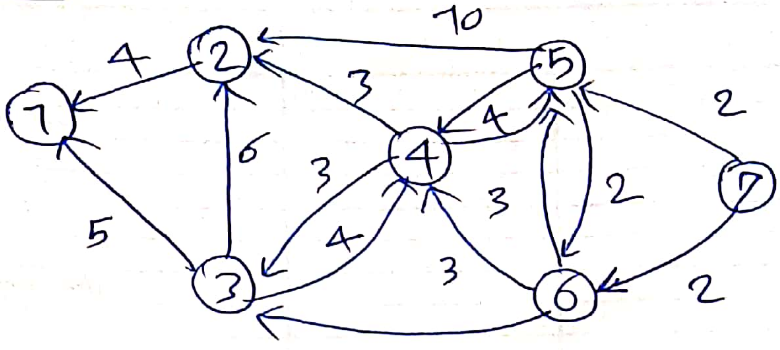
جواب:

min = مقادیر ستون آخر

جواب = $\min\{6, 7, \infty\} = \boxed{6}$

سوال ٤

درخت کوتاه‌ترین مسیر از هر گره به گره‌ی ١ برای گراف زیر به کمک Bellman Ford



الگوریتم Bellman Ford برای پیدا کردن کوتاه‌ترین مسیر از یک مبدأ (src) به تمام راس‌ها داخل گراف است ← برای پیدا کردن کوتاه‌ترین مسیر از هر راس به یک راس خاص 6 باید هر بار 6 راس های دیگر بجز راس ① را src ها بگیریم و الگوریتم را اجرا کنیم تا کوتاه‌ترین مسیر به راس ١ پیدا شود ←

① src = 7 ⟶ تا گام ۵ جدول (V−1) تا گام — 7 تا راس داریم

	0	1	2	3	4	5
7	0	0	0	0	0	0
6	∞	2	2	2	2	2
5	∞	2	2	2	2	2
4	∞	∞	5	5	5	5
3	∞	∞	11	8	8	8
2	∞	∞	12	8	8	8
1	∞	∞	∞	16	12	12

$$L_{i,v} = \min \begin{cases} L_{i-1,v} \\ \min\{L_{i-1,w} + P_{wv}\} \end{cases}$$

مثلا

$$A_{2,2} = \min\{\infty, \{\underbrace{A_{1,4}}_{\infty}+3\}, \{\underbrace{A_{1,3}+6}_{\infty}\}, \{\underbrace{A_{1,5}+10}_{12}\}\}$$

$$\boxed{A_{2,2} = 12}, \ldots$$

جدول پیدا کردن مسیر

	0	1	2	3	4	5
7	—	—	—	—	—	—
6	—	7	7	7	7	7
5	—	7	7	7	7	7
4	—	—	6	6	6	6
3	—	—	6	4	4	4
2	—	—	5	4	4	4
1	—	—	—	2	2	2

← parent home

7 → 6 → 4 → 2 → 1

مسیر از 7 به 1

Scanned by CamScanner

$\boxed{src = 6}$ → $v-1 =$ س

7	∞	∞	∞	∞	∞
6	0	0	0	0	0
5	∞	3	3	3	3
4	∞	3	3	3	3
3	∞	9	6	6	6
2	∞	∞	6	6	6
1	∞	∞	14	10	10
	0	1	2	3	4

با الم جدول parent

مسیر از v=6 به v=1 :

6 → 4 → 2 → 1

$\boxed{src = 5}$

7	∞	∞	∞	∞	∞	∞
6	∞	2	2	2	2	2
5	0	0	0	0	0	0
4	∞	6	5	5	5	5
3	∞	∞	9	8	8	8
2	∞	∞	9	8	8	8
1	∞	∞	14	13	12	12
	0	1	2	3	4	5

مسیر از v=5 به v=1 :

5 → 6 → 4 → 2 → 1

$\boxed{src = 4}$

مسیر از v=4 به v=1 :

4 → 2 → 1

7	∞	∞	∞	∞
6	∞	∞	6	6
5	∞	4	4	4
4	0	0	0	0
3	∞	3	3	3
2	∞	3	3	3
1	∞	∞	7	7
	0	1	2	3

$\boxed{src = 2}$

7	∞	∞	∞
6	∞	∞	∞
5	∞	∞	∞
4	∞	∞	∞
3	∞	∞	∞
2	0	0	0
1	∞	4	4
	0	1	2

مسیر از v=2 به v=1 :

$\boxed{2 \to 1}$

$\boxed{src = 3}$

کوتاه‌ترین مسیر از v=3 به v=1 :

$\boxed{3 \to 1}$

	0	1	2	3	4
7	∞	∞	∞	∞	∞
6	∞	∞	∞	10	10
5	∞	∞	8	8	8
4	∞	4	4	4	4
3	0	0	0	0	0
2	∞	6	6	6	6
1	∞	5	5	5	5

i	c_i^1	c_i^2	c_i^3	c_i^4	c_i^5	shortest path
1	0	0	0	0	0	X
2	4	4	4	4	4	2→1
3	5	5	5	5	5	3→1
4	∞	7	7	7	7	4→2→1
5	∞	14	13	12	12	5→6→4→2→1
6	∞	14	10	10	10	6→4→2→1
7	∞	∞	16	12	12	7→6→4→2→1

سوال 5

۸ قاچاقچی با شماره‌ی ۱ تا ۸

هدف جابجایی از ۱ به ۸ . هزینه‌ی رشوه از i به j ← $\boxed{c_{ij}}$

i تنها سه دفعه می‌تواند گذر کند . کمترین هزینه = e

راه حل e از الگوریتم Bellman Ford می‌توانیم استفاده کنیم .

c_{ij} ← وزن یال‌ها از i به j * افراد ۱ تا ۸ ← راس‌ها در گراف

پس یک آرایه به اسم ans به صورت ans[8][8] تعریف می‌کنیم که در ans[i][j] کمترین هزینه‌ی لغد از قاچاقچی ۱ به j است . که ۲ یال وجود دارد

اگر جواب نداشتیم ← ∞ در آن ذخیره شود.

ans[0][1] = 0 ans[0][j] = 0
$1 < j \leq 8$

$$ans[i][j] = \min\{ans[i-1][j] \,,\, \min\{ans[i-1][k] + c_{kj}\}\}$$

مثل الگوریتم های قبلی، کمترین هزینه ← رفتن از k به j

جواب → min { مستقیم با i-1 یال به j برویم یا
min { حالاتی که به یک قاچاقچی به نام k برویم و از k به j برویم } }

جواب در اینجا → در ans[8][8] قرار دارد.

از الگوریتم Bellman Ford استفاده کردیم ← Time complexity = $\boxed{O(n^3)}$

ارائه‌ی 5

چندجمله‌ای

اما طبق توضیح جزو الگوریتم Bellman Ford ، می‌توانیم بهینه‌تر کنیم . چون ممکن است در جدول به ستون‌های تکراری برسیم ← برای افزایش بهره‌وری می‌توانیم یک متغیر تعریف کنیم که هر وقت دیگر راسی تغییر نکرد، برنامه تمام شود

در این صورت ← از order $\boxed{O(nm)}$ می‌شود

سوال ۴ $\boxed{n \text{ شهر}}$ بین هر دو شهر ← جاده

- اولی ← شهر اول به دوم
- دومی ← شهر دوم به اول

t_{ij} = مدت زمانی که در مسیر i به j حرکت کنیم

$t_{ij} \neq t_{ji}$

کوتاه‌ترین مسیر از شهر i به j ؟

با الگوریتم Floyd - warshall

راه حل θ اول i و j های مربوط به شهرهای تصرف شده را حذف می‌کنیم

و سپس تعداد شهرهایی که هنوز تصرف نشده‌اند را $\boxed{n}$ فرض می‌کنیم.

آرایه‌ی سه بعدی $\boxed{ans}$ را به صورت ans[n+1][n][n] درنظر می‌گیریم

ans[k][i][j] ← کمترین زمان برای رفتن از i به j با عبور از $\boxed{k}$

$$ans[0][i][j] \rightarrow \begin{cases} \text{if } i=j \rightarrow 0 \\ \text{if } i,j \in E \rightarrow t_{ij} \\ \text{if } i,j \notin E \rightarrow +\infty \\ \text{if } i \neq j \rightarrow +\infty \end{cases}$$

ans[k][i][j] =

$$ans[K][i][j] = \min \begin{cases} ans[K-1][i][j] \\ ans[K-1][i][K] + ans[K-1][K][j] \end{cases}$$

K = یک شهر بین شهرها

برای پیدا کردن کوتاه‌ترین مسیر از i به j یک شهر مثل K را به عنوان شهر میانی انتخاب می‌کنیم

و کمترین مسیر را در آرایه‌ی $\boxed{ans}$ ذخیره می‌کنیم