دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

# تکلیف سوم درس طراحی الگوریتم‌ها

نیم‌سال تحصیلی: بهار ۱۴۰۲

مدرّس: دکتر محمّدرضا حیدرپور

دستیاران آموزشی: مصطفی دریس‌پور - مجید فرهادی - محمّدیاسین کرباسیان - محمّدرضا مزروعی - امیر منصوریان - امیرارسلان یاوری

## ۱ اسکله‌های رودخانه

رودخانه‌ای دارای $n$ اسکله با شماره‌های $1$ تا $n$ مفروض است. اگر هزینه اجاره قایق برای رفتن از اسکله $i$ام به اسکله $j$ام برابر $a_{ij}$ باشد، الگوریتمی با رویکرد برنامه‌نویسی پویا برای محاسبه کمینه هزینه برای رفتن از اسکله $1$ام به اسکله $n$ام ارائه دهید. (۱۰ نمره)

## ۲ تاس‌های چندوجهی

$n$ تاس $m$وجهی مفروض است. الگوریتمی با رویکرد برنامه‌نویسی پویا برای شمارش تمام حالاتی که مجموع تاس‌ها برابر $X$ است ارائه دهید. (۱۰ نمره)

## ۳ بزرگ‌ترین زیرماتریس مربعی

یک ماتریس شامل اعداد صفر و یک مفروض است. الگوریتمی با رویکرد برنامه‌نویسی پویا برای یافتن اندازه بزرگ‌ترین زیرماتریس مربعی که تمام عناصر آن یک است ارائه دهید. (۲۰ نمره)

$$Array = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \implies Order = 2$$

## ۴ اندیس‌های آرایه

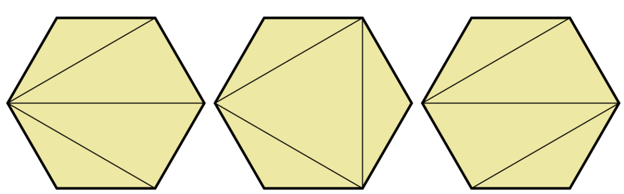
آرایه $A$ مفروض است. الگوریتمی با رویکرد برنامه‌نویسی پویا برای یافتن اندیس‌های $p$، $q$، $r$ و $s$ از این آرایه با شرط $s > r > q > p$ به طوری که مقدار $A[s] - A[r] + A[q] - A[p]$ بیشینه باشد ارائه دهید. (۲۰ نمره)

## ۵ کوئرا

به دو سوال از سه سوال کوئرا پاسخ دهید. (۴۰ نمره)

## ۶ وترهای کمینه

مختصات رئوس یک $n$ضلعی محدب مفروض است. الگوریتمی با رویکرد برنامه‌نویسی پویا برای تقسیم این $n$ضلعی به $n-2$ مثلث به طوری که هیچ دو وتری یک‌دیگر را قطع نکنند و مجموع طول وترها کمینه باشد ارائه دهید. (۲۰ نمره مازاد)



✓

① فرض می کنیم که همه ی اسکله ها به هم راه دارند حال:

اگر $dp[i][j]$ را برابر با min هزینه برای رسیدن از اسکله $i$ به اسکله $j$ با شرط استفاده از اسکله های میان این ۲ یعنی اسکله های میان $i$ و $j$ بدانیم، داریم:

$dp_{i,j}$ : min هزینه برای رفتن از $i$ به $j$ با شرط استفاده از اسکله های میان $i,j$

بدیهی است که:

$dp_{i,j} = a_{i,j}$ (همگی به همراه دارند)

$dp_{i,j} = 0 \qquad i = j$

$dp_{i-1,i} = a_{i-1,i}$ (هیچ اسکله ای میان این ۲ نیست)

فرمول بازگشتی:

$dp_{1,1} = 0$

رفتن به $n-k$ و سپس از $k$ به $n$

$$dp_{i,j} = \min\left(dp_{i,j}\ ,\ dp_{i,k} + a_{k,j}\right)$$

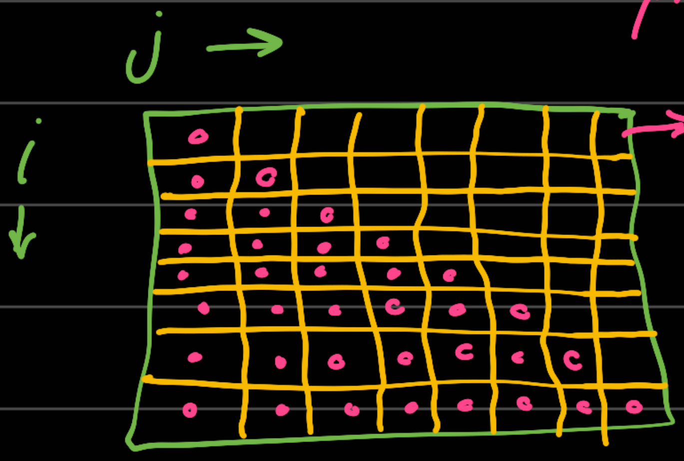
$k$ یک اسکله میان $i,j$ است. الگوریتم اینگونه کار می کند که برای رسیدن از ۱ به $n$ ما فرض می کنیم مقدار min برای رسیدن از ۱ به $n-k$ را داریم حال برای رسیدن به $n$ ۲ راه داریم:

① از 1 به $n-k$ برویم سپس از $n-k$ به $n$ برویم.

② مستقیما از 1 به $n$ برویم.

در هر مرحله میاییم این ۲ مقدار را min محاسبه می کنیم و درون $dp_{i,j}$ قرار می دهیم.

پاسخ نهایی ما $dp_{1,n}$ است به این معنی که کمترین هزینه برای رفتن از اسکله اول به اسکله $n$ ام.




$\rightarrow dp_{1,n} \rightarrow$ پاسخ ما

این الگوریتم از $O(n^3)$ می باشد چرا که ۳ حلقه $n$ تایی دارد (۱ برای $i$، و ۲ برای $j$ و ۳ برای $k$) (به نوعی همان الگوریتم فلوید وارشال است)

② یک آرایه ی ۲ بعدی به نام $dp_{i,j}$ داریم

فرض کنید $dp_{i,j}$ تعداد روش ها برای رسیدن به مجموع $j$ با استفاده از $i$ تاس اول باشد. پس برای رسیدن به مجموع $x$ با استفاده از $n$ تاس می توانیم یک تاس جدید با مقدار 1 تا $m$ اضافه کنیم و آن را با $i-1$ تاس قبلی جمع کنیم تا به مقدار $x$ (همان $j$) برسیم

با توجه به توضیحات فوق به فرمول بازگشتی زیر می رسیم:

$$dp_{i,j} = dp_{i-1,j-1} + dp_{i-1,j-2} + \cdots + dp_{i-1,j-m}$$

دقت شود که $m \leq n$ می باشد. همچنین مثلا با ۳ تاس ۴ وجهی نمی توان مجموع ۱۳ را ساخت.

برای محاسبه خانه ی $dp_{i,j}$ فقط به اعضای ردیف قبلی نیاز داریم.

توجه شود که حالت پایه ی ما برابر با $dp_{0,0} = 0$ می باشد و باید بقیه ی خانه ها را صفر کنیم. (چون جمع داریم، garbage_value کار را خراب می کند)

پاسخ نهایی ما درون خانه $dp_{max}$ می باشد.

③ مانند سوال دوم کوئرا می باشد منتها زیر ماتریس مربعی با درایه های یک را می خواهیم:

یک آرایه ۲ بعدی به نام $dp_{i,j}$ داریم. اولین ردیف وستون $dp$ را مقادیر خود ماتریس پر می کنیم سپس روی تک تک عناصر $A$ حرکت می کنیم اگر یک بود خانه های (بالا - چپ - بالا چپ) را $min$ گرفته و به علاوه ی یک می کنیم و درون $dp_{i,j}$ می ریزیم. اگر ماتریس داده شده $m \times n$ باشد $dp$ ما $(m+1) \times (n+1)$ است.

فرمول بازگشتی :

if $A_{i,j} = 1$ :

$$dp_{i,j} = 1 + \min(dp_{i-1,j}, dp_{i,j-1}, dp_{i-1,j-1})$$

بالا     چپ     بالا چپ

با یک بار گشتن روی dp و پیدا کردن max ما به پاسخ سوال می رسیم (می توان این کار را هنگام پر کردن همزمان انجام داد)

این الگوریتم از $O(n \times m)$ می باشد.

(4) برای اینکه مقدار $A_s - A_r + A_q - A_p$ max باشد باید

مقادیر $A_p$ و $A_r$ min مقادیر A همچنین $A_q$ و $A_s$ بزرگترین مقادیر A باشند. سپس مجموعه جواب A ها را طوری انتخاب می کنیم که شرط $s > r > q > p$ برقرار شود ← حالت

brute force

brute force : فرض می کنیم q و r و s جواب نهایی را داریم حال p را به گونه ای پیدا می کنیم که شرایط مسئله مهیا شود.

حال فرض می کنیم r و s را داریم و p را نیز قبلا یافته‌ایم، حال q را به گونه ای میابیم که شرایط برقرار باشد.

همینطور مراحل بالا را برای r و s عملی می کنیم تا s و r را بیابیم.

DP : ایده کلی روی محدود کردن p می باشد چون p > q > r > s

۲ آرایه با اندازه len(A) با نام های left و right .

left[0] = A[0] و روی A حرکت می کنیم و هربار

$$left_i = max(A_i , A_i + left_{i-1})$$

right[n-1] = A[n-1] روی A از راست به چپ حرکت می کنیم و هربار

$$right_i = max(A_i , A_i + right_{i+1})$$

۴ متغیر maxdiff و maxr و maxq و maxp را برابر با $-\infty$ قرار می دهیم . از چپ به راست روی A حرکت می کنیم و داریم :

```
for i in range(n-۲):
    for j in range(i+۲, n):
        for k in range(i+۱, j):
            ans = A_j + A_i + A_k - A_maxp + left_(k-1) + right_(j+1)
            if ans > maxdiff:
                maxdiff = ans
                maxr = i
                maxq = k
                maxp = maxindex(right[j+1:n])
                maxs = j
```

و در نهایت پس از پایان حلقه ها جو pd و ۲ پیدا شده اند

تابع maxindex ایندکس بزرگترین آرایه داده شده بهش را return می کند.

6

min طول قطرها برای رأس j-i : $D[i][j]$

$i < j$

حالت پایه برای $j=i+2$ و $i=1$ می باشد یعنی یک مثلث داریم که خب تعداد قطرهای آن صفر و $dp_{1,3} = 0$ می باشد.
برای پر کردن dp را k داریم که از $i+2$ شروع می‌کند و ۲ تا ۲ تا تا $j-1$ می رود.

رابطه بازگشتی: $dp_{ij} = \min(dp_{ij}, dp_{ik} + dp_{kj} + dis(i,j))$

$i > j > k \quad j = i+2$

که در آن تابع dis فاصله میان i و j را برمی‌گرداند.
خانه‌ی $dp_{1,n}$ پاسخ نهایی مسئله ما می باشد. این الگوریتم از $O(n^3)$ می باشد