جلسه دهم

Sunday, April 4, 2021 13:07

موضوع: برنامه نویسی پویا - مساله ضرب زنجیری ماتریس و مساله کوله‌پشتی ۰-۱

ضرب ماتریسی: فرض کنید A یک ماتریس $n \times m$ باشد و B یک ماتریس mp باشد.

$A \times B$ چه ابعادی دارد؟ $n \times p$

* $n \times m \times p$: $A \times B$

فرض کنید ۴ عدد ماتریس داریم:

$$B = A_1 \times A_2 \times A_3 \times A_4$$

10×100 $\quad 100 \times 5$ $\quad 5 \times 50$ $\quad 50 \times 1$ → 10×1

ضرب ماتریس یک عمل اشتراک پذیر است.

$10\times100 \quad 100\times5 \quad 5\times1$

- $(A_1(A_2 \times (A_3 \times A_4)))$: $250 + 500 + 1000 = 1750$
- $((A_1 \times A_2) \times (A_3 \times A_4))$: $5000 + 250 + 50 = 5300$

$10\times5 \quad 5\times1$

- $(((A_1 \times A_2) \times A_3) \times A_4)$: $5000 + 2500 + 500 = 8000$

$10\times\cancel{5}\ \cancel{5}\times50 \quad 50\times1$

تعداد روش های مختلف پرانتزگذاری؟ ← عدد n ام کاتالان: $\Omega(4^n / n^{3/2})$

هدف، پیدا کردن پرانتزبندی است که تعداد ضرب ها را کمینه کند.

صورت مساله: n عدد ماتریس $A_1, A_2, \dots, A_n$ داده شده است.

$P_1 \times P_2 \quad P_2 \times P_3 \quad P_n \times P_{n+1}$

ماتریس A_i، دارای ابعاد $P_i \times P_{i+1}$ است. هدف پیدا کردن پرانتزبندی بهینه برای ضرب $A_1 \times A_2 \times \dots \times A_n$ است.

* ایده؟

- * در هر مرحله کمترین P_{i+1} را پیدا و آن را ضرب کنیم
- * در هر مرحله دو ماتریس با کمترین هزینه را در هم ضرب کنیم
- *

روش پویا

$$\boxed{A_1} \times (A_2 \times A_3 \times \dots \times A_n) \rightarrow$$

$(A_1$

ایده: فرض کنید یک $\boxed{\text{Oracle}}$ به ما می‌گوید که آخرین عملیات، ضرب k ام است.

$$\underbrace{B_{1-n}} = \underbrace{(A_1 \times A_2 \times A_3 \times \dots \times A_k)}_{B_{1-k}} \times \underbrace{(A_{k+1} \times \dots \times A_n)}_{B_{k+1-n}} + P_1 \times P_{k+1} \times P_{n+1}$$

$P_1 \times P_2$ $\quad P_2 \times P_3 \quad \dots$

$P_1 \times P_{k+1}$ $\quad\quad P_{k+1} \times P_{n+1}$

$A_1 \times A_2 \times A_3 \times \dots \times A_n$

حالا که k را نمی‌دانیم، همه حالات را چک می‌کنیم!

$$B_{1-n} = \min_{k:1 \to n-1} B_{1-k} + B_{k+1-n} + P_1 \times P_{k+1} \times P_{n+1}$$

$$\boxed{A_1} \times A_2 \times A_3 \times A_4 \times A_5 \longrightarrow \quad A_1 \quad A_2 \quad A_3 \quad A_4 \quad A_5$$

for $i:1 \to 5$
    for $j: i+1 \to 5$

B_{i-j} : $B[i][j]$

$i \xleftrightarrow{\ l\ } j \to i+l-1$

5

1 — 5

2 — 6

3 — 7

```
for(i:1 -> n)
    B[i][i] = 0
for(l:2 -> n)
    for(i:1 -> n-l+1)
        j = i+l-1
        B[i][j] = ∞
        for(k:i -> j)
            if B[i][k] + B[k+1][j] + P_i P_{k+1} P_{j+1} < B[i][j]
```

ها for (k: i → j)

if $\underbrace{B[i][k] + B[k+1][j] + P_i P_{k+1} P_{j+1}} < B[i][j]$

$B[i][j] =$

زمان اجرا : $O(n^3)$.

* مساله کوله پشتی ۰-۱ :

n شیئ در مکانی وجود دارد. شیی i ام حجم v_i و ارزش w_i دارد. همچنین یک کوله‌پشتی داریم که حجم آن برابر V است. می‌خواهیم یک زیرمجموعه از اشیا را انتخاب کنیم و در کوله قرار دهیم به طوری حجم آن ها حداکثر V و ارزش آن ها بیشینه شود.

بیشینه: «دزدی» :

مثال : V = ۱۰

حجم	۵	~~۵~~	۳	۴	~~۷~~	۱
ارزش	۵	۳	۲	۶	۸	۱
				۴٫۵		۱۲

* ایده : اشیا را برحسب نسبت ارزش به حجم مرتب کنیم و به آن ترتیب انتخاب کنیم

حجم	۱	۱۰
ارزش	۲	۱۵

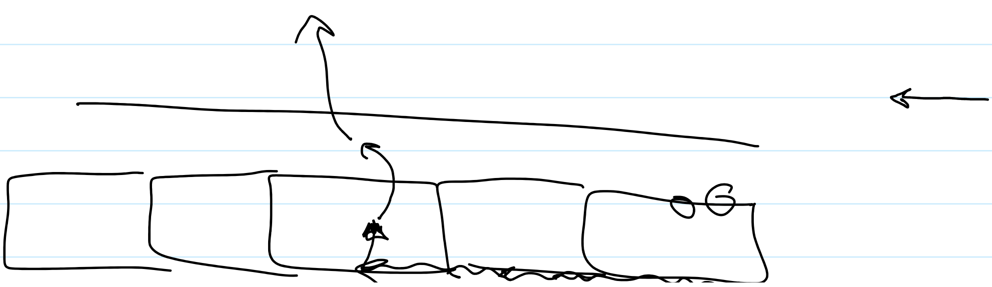
$\rightarrow \boxed{۱۰}$

۲

۹ ۱۰ ۱۰

* چه زمانی روش حریصانه درست جواب می‌داد؟

{ fractional knapsack ← زمانی که اشیا divisible باشند.



6