دانشکده‌ی مهندسی کامپیوتر

یادآوری جلسه دهم — برنامه‌نویسی پویا

در جلسه قبل در ادامه بحث برنامه‌نویسی پویا، با مساله ضرب زنجیری ماتریسی و همچنین مساله کوله‌پشتی آشنا شدیم. در ادامه این دو مساله را به طور مختصر یادآوری می‌کنیم.

مسئله ضرب زنجیری ماتریس‌ها

در این مساله، یک دنباله از ماتریس‌ها به صورت $A_1, A_2, \ldots, A_n$ داده شده است. ابعاد ماتریس i ام به صورت $p_i \times p_{i+1}$ می‌باشد. هدف، ارائه یک پرانتزبندی برای محاسبه عبارت $A_1 \times A_2 \times A_3 \times \ldots \times A_n$ به گونه‌ای است که میزان ضرب‌های استفاده شده را کمینه کند. برای محاسبه ضرب یک ماتریس $x \times y$ در یک ماتریس $y \times z$ نیاز به xyz ضرب است و خروجی آن یک ماتریس $x \times z$ خواهد بود.

برای حل مساله ضرب زنجیری ماتریس‌ها با استفاده از برنامه نویسی پویا، $B[i][j]$ را برابر با کم‌ترین تعداد ضرب لازم جهت محاسبه ضرب زنجیره ماتریس‌ها از ماتریس i ام تا ماتریس j ام در نظر بگیرید. در این صورت، هدف محاسبه $B[1][n]$ است. حال اگر بدانیم که آخرین ضرب محاسبه شده ضرب k ام باشد، در این صورت میزان $B[1][n]$ را از طریق رابطه بازگشتی زیر می‌توان حساب کرد: $B[1][n] = B[1][k] + B[k+1][n] + p_1 p_{k+1} p_{n+1}$. حال چون از این که کدام ضرب آخر است مطلع نیستیم، می‌توانیم بین تمام انتخاب‌های ممکن برای k مقدار کمینه را انتخاب کنیم.

$$B[1][n] = \min_k \quad B[1][k] + B[k+1][n] + p_1 p_{k+1} p_{n+1}.$$

Algorithm 1: MCM

```
for i ← 1 to n do
    B[i][i] ← 0
for l ← 2 to n do
    for i ← 1 to n − l + 1 do
        j ← i + l − 1
        B[i][j] ← ∞
        for k ← i to j do
            if B[i][k] + B[k + 1][j] + p_i p_{k+1} p_{j+1} < B[i][j]
            then
                B[i][j] ← B[i][k] + B[k+1][j] + p_i p_{k+1} p_{j+1}
return B[1][n]
```

الگوریتم ۱ پیاده‌سازی پویای این رابطه بازگشتی را با زمان $\mathcal{O}(n^3)$ نشان می‌دهد.

مساله کوله‌پشتی ۰ ـ ۱

فرض کنید یک کوله‌پشتی با حجم V داریم. همچنین n شی داریم که شی iام دارای حجم v_i و ارزش w_i است. هدف، انتخاب یک زیرمجموعه از اشیا است، به گونه‌ای که مجموع حجم آن‌ها حداکثر V شود و همچنین مجموع ارزش اشیای انتخاب شده بیشینه شود. به عنوان نمونه، اگر ۴ شی با ارزش‌های ۱۰، ۳۰، ۶۰ و ۴۰ و وزن‌های ۲، ۴، ۵ و ۳ داشته‌باشیم و حجم کوله‌پشتی ۱۰ باشد، بهترین کار انتخاب اشیای ۱، ۳ و ۴ است.

دقت کنید که تمام اعداد این سوال اعداد طبیعی هستند. برای حل این سوال، ابتدا روش حریصانه را که در هر مرحله شی با بیشترین نسبت ارزش به حجم که قابل انتخاب است را انتخاب کنیم، مورد بررسی قرار دادیم، و نشان دادیم که این روش لزوما به انتخاب بهینه نمی‌انجامد. به عنوان نمونه فرض کنید یک کوله با حجم ۱۰ و دو شی با حجم‌های ۱ و ۱۰ و ارزش‌های ۲ و ۱۰ داریم. در این صورت الگوریتم حریصانه ارزش ۲ و جواب بهینه ارزش ۱۰ را تضمین می‌کند.

پرسش اگر برای مساله کوله پشتی، اگر $opt[i][j]$ را جواب بهینه برای زیرمسأله شامل n شی اول و حجم باقی‌مانده j در نظر بگیریم، یک رابطه بازگشتی برای محاسبه $opt[i][j]$ ارائه دهید.

پاسخ های خود را می توانید تا قبل از شروع کلاس به این لینک ارسال کنید.