دانشکده‌ی مهندسی کامپیوتر

یادآوری جلسه یازدهم — برنامه‌نویسی پویا — گردآورنده: محمدحسین حاجی سیدسلیمان ـ زهرا فاضل

در جلسه قبل در ادامه برنامه‌نویسی پویا با مسئله کوله‌پشتی آشنا شدیم. در این مسئله، کوله‌ای به حجم V و مجموعه‌ای از n شی داریم که شی iام دارای ارزش w_i و حجم v_i است. می‌خواهیم یک زیرمجموعه از اشیاء با بیشینه مجموع ارزش و حجم حداکثر V انتخاب کنیم. این مسئله دو مدل دارد: مسئله کوله‌پشتی ۱ ـ ۰ و مسئله کوله‌پشتی کسری. همچنین با مسئله مجموع زیرمجوعه‌ها آشنا شدیم.

مسئله کوله‌پشتی ۱ ـ ۰

در این مدل از مسئله کوله‌پشتی یا یک شی به طور کامل انتخاب می‌شود و یا کنار گذاشته‌می‌شود و نمی‌توان بخشی از یک شی را انتخاب کرد. در جلسات قبل برای این مسئله الگوریتم حریصانه ارائه کردیم و نشان دادیم که این الگوریتم الزاما پاسخ بهینه را بر نمی‌گرداند. حال برای این مسئله راه‌حل برنامه‌نویسی پویا ارائه می‌کنیم.

فرض کنیم $opt[i][j]$ برابر بیشترین ارزش ممکن برای مسئله کوله‌پشتی به ازای i شی اول و کوله به حجم j باشد. اگر شی iام در $opt[i][j]$ انتخاب نشود، آنگاه $opt[i][j]$ برابر $opt[i-1][j]$ و اگر انتخاب شود، آنگاه $opt[i][j]$ برابر $opt[i-1][j-v_i]+w_i$ است. بنابراین $opt[i][j]$ به صورت زیر به شکل بازگشتی قابل محاسبه است.

$$opt[i][j] = \max(opt[i-1][j], opt[i-1][j-v_i]+w_i)$$

Algorithm 1: 0-1 Knapsack

```
for k ← 1 to n do
    opt[k][0] ← 0
    opt[0][k] ← 0
for i ← 1 to n do
    for j ← 1 to V do
        opt[i][j] = max(opt[i-1][j], opt[i-1][j-v_i] + w_i)
return opt[n][V]
```

الگوریتم ۱ پیاده‌سازی پویای این رابطه بازگشتی را با زمان $O(nV) = O(n2^{\log V})$ نشان می‌دهد. این مسئله NP-hard است که راجع به این موضوع در جلسات آتی صحبت خواهیم کرد.

مسئله مجموع زیرمجموعه‌ها

مجموعه‌ای شامل n عدد $x_1, x_2, \ldots, x_n$ داریم. می‌خواهیم بدانیم که آیا زیرمجموعه‌ای از آن با مجموع k شود.

فرض کنیم $f[i][j]$ یک مقدار $Boolean$ و برابر پاسخ «آیا زیرمجموعه‌ای از i عدد اول با مجموع j وجود دارد؟» باشد. در این صورت رابطه بازگشتی روبه‌رو برقرار است.

$$f[i][j] = \begin{cases} f[i-1][j] & x_i \text{ is not chosen.} \\ f[i-1][j-x_i] & x_i \text{ is chosen.} \end{cases}$$

پاسخ برابر $f[n][k]$ و پیچیدگی زمانی الگوریتم $O(nk)$ است.

فرم برنامه‌نویسی پویای این مسئله مشابه فرم برنامه‌نویسی پویای مسئله کوله‌پشتی ۱ ـ ۰ است و می‌توان به آن‌ها را متناظر کرد. به این صورت که متناظر با مسئله کوله‌پشتی ۱ ـ ۰ است که حجم کوله k باشد و x_i ارزش و حجم شی iام است.

الگوریتم تقریبی برای مسئله کوله‌پشتی

در بخش انتهای کلاس، یک الگوریتم تقریبی برای مسئله کوله‌پشتی ۱ ـ ۰ ارائه دادیم. الگوریتم پیشنهادی به شرح زیر است:

۱. اشیا را بر حسب نسبت ارزش به حجم آنها مرتب کنیم.

۲. در هر مرحله بین اشیای انتخاب‌نشده، شی با بیشترین نسبت ارزش به حجم را انتخاب می‌کنیم تا زمانی که نتوان شی دیگری انتخاب کرد. مجموعه اشیای انتخاب‌شده را A می‌نامیم و فرض کنیم $k = |A|$.

۳. مجموعه B را برابر شی $k+1$ام قرار می‌دهیم.

۴. بین مجموعه A و B، پاسخ بهتر را برمی‌گردانیم.

این الگوریتم در زمان $O(n \log n)$ پاسخی با تقریب $\frac{1}{2}$ برای مسئله کوله‌پشتی ۱ ـ ۰ برمی‌گرداند. برای اثبات تقریب در نظر داشته باشید که مجموعه اشیای $A \cup B$ در مجموع ارزش بیش از پاسخ مسئله کوله‌پشتی در حالت کسری را تولید خواهند کرد و این پاسخ نیز حد بالایی برای کوله‌پشتی ۱ ـ ۰ است.