دانشکده‌ی علوم ریاضی

آنالیز الگوریتم‌ها ۱۹ اسفند ۹۱

جلسه‌ی ۱: ۱۹: مسائل کلاس NP ـ کامل

مدرّس: دکتر شهرام خزائی نگارنده: سید مهدی میرکیا

۱ مقدمه

در جلسات قبل با کلاس‌های مختلف مسائل آشنا شدیم. یکی از این کلاس‌ها، کلاس مسائل NPـکامل [۱] بود. مسائل متعلق به این کلاس، سخت‌ترین مشائل برای پاسخگویی هستند از آن جهت که برای آنها هیچ الگوریتم شناخته شده‌ی قابل اجرا در زمان چند جمله‌ای وجود ندارد. در ادامه به بررسی روش‌های برخورد با این مسائل می‌پردازیم.

۲ روش‌های برخورد با مسائل NPـکامل

مواجه شدن با این دسته از مسائل به منزله‌ی پایان راه نیست. انتخاب یک روش و استراتژی مناسب برای برخورد با آنها، می‌تواند منجر به راه‌حلی قابل قبول شود. از جمله‌ی این استراتژی‌ها می‌توان به سه روش زیر اشاره کرد:

۱.۲ استفاده از روش‌های فرا ابتکارانه و محاسبه‌ی جواب تقریبی

مبنای این روش، پیشگرفتن یک استراتژی فرا ابتکارانه(مانند روش‌های حریصانه [۲] یا برنامه‌ریزی پویا [۳]) می‌باشد. استفاده از الگوریتم‌هایی که سریع‌تر ما را به جواب می‌رسانند اما همواره صحیح نیستند.

مثال ۱ *مسئله‌ی پوشش مجموعه [۴] که با انتخاب یک روش حریصانه تقریبی، منجر به رسیدن به جواب در پیچیدگی زمانی $klnn$ می‌شود.*

۲.۲ تمرکز روی حالت‌های خاص

هنگامی که اندازه‌ی جواب بهینه کوچک باشد، می‌توان با تکیه بر آن، روشی طراحی کرد که منجر به پاسخ نهایی مد نظر باشد.

مثال ۲ *پیدا کردن کوتاه‌ترین مسیر در گراف بدون دور منفی، و یا پیدا کردن بزرگ‌ترین مجموعه‌ی مستقل [۵] در درخت‌ها*

[۱] NP-Complete
[۲] Greedy
[۳] Dynamic Programming
[۴] Set Cover
[۵] Independent Set

مثال ۳ *در مساله‌ی کوله‌پشتی نیز، در حالتی که $w = O(n)$ باشد، با یک حالت خاص مواجهیم که می‌توان در پیچیدگی زمانی $O(nw)$ به پاسخ نهایی دست یافت.*

مثال ۴ *در مسئله‌ی پوشش راسی، حالت $k = O(logn)$ یک حالت خاص محسوب می‌شود.*

۳.۲ راه‌حل‌های دقیق

گاهی با مسائلی روبه‌رو هستیم که می‌توان در زمانی نمایی به پاسخی برای آن‌ها دست یافت. این روش با وجود زمان‌گیر بودن، بر حالت استفاده از جست‌وجوی خام و بی‌خردانه [۶] ارجحیت دارد.

مثال ۵ *تبدیل زمان پاسخ‌گویی به مسئله‌ی فروشنده‌ی دوره‌گرد از پیچیدگی زمانی $(O(n!))$ به $O(n^{2}2^{n})$*

مثال ۶ *در مسئله‌ی کوله‌پشتی، تبدیل روشی با پیچیدگی زمانی $O(2^{n})$ به $O(nw)$.*

۳ مساله‌ی پوشش راسی

مساله‌ی پوشش راسی [۷] را به شکل زیر در نظر بگیرید:
ورودی: گراف $G = (V, E)$ با n راس و عدد صحیح k .
خروجی: مجموعه‌ی $S \subseteq V$ با اندازه‌ی k که حداقل یکی از دو سر هر یال گراف در S واقع باشد.

۱.۳ راه‌حل ۱

حالتی را در نظر می‌گیریم که در آن $k = O(logn)$ باشد. با جست‌وجوی تمام حالات ممکن، به پاسخی از پیچیدگی زمانی $O(n^{k})$ می‌رسیم. آیا راه‌حل بهینه‌تری وجود دارد؟

۲.۳ راه‌حل ۲

یال uv از گراف داده شده را در نظر می‌گیریم. یک‌بار u را با تمامی یال‌های متعلق به آن حذف کرده تا گرافی مانند G_u به‌دست آید، و باری دیگر این عمل را برای راس v تکرار می‌کنیم. این روش، در پیچیدگی زمانی $O(2^{k}m) = O(mn)$ ما را به پاسخ می‌رساند. در ادامه به بررسی دقیق‌تر این روش می‌پردازیم.

۳.۳ لم زیر مسئله‌ی بهینه

قضیه ۱ *G دارای یک پوشش راسی k عضوی است، اگر و فقط اگر G_u یا G_v دارای پوشش راسی $k - 1$ عضوی باشد.*

[۶] Brute-force Search
[۷] Vertex Cover

برهان. طرف اول اثبات:
فرض کنید G_u دارای پوشش راسی مانند S با اندازه‌ی $k-1$ باشد. در نظر می‌گیریم: $E = E_u \cup F_u$ از آنجایی که S شامل رئوسی از هر یال موجود در E_u می‌باشد، $S \cup \{u\}$ یک پوشش راسی با اندازه‌ی k برای G خواهد بود.
برای اثبات طرف دوم داریم:
فرض کنید $S = a$ یک پوشش راسی با اندازه‌ی k برای گراف G باشد. از آنجایی که (u, v) یک یال برای G است، حداقل یکی از آن‌ها مانند u عضو مجموعه‌ی S است. از طرفی $S - \{u\}$ نیز باید یک پوشش راسی با اندازه‌ی $k-1$ برای G_u باشد، چرا که هیچ‌یک از یال‌های E_u در راس u تلاقی ندارند.
■

۴.۳ الگوریتم بهینه برای مساله‌ی پوشش راسی

Algorithm 1 Algorithm: A Search Algorithm

```
function SEARCH(undirected graph G = (V, E), integer k)
    Ignore base cases
    Pick an arbitary edge (u, v) ∈ E
    Recursively search for a vertex cover S of size (k − 1) in G_u
    if found then
        return S ∪ {u}
    Recursively search for a vertex cover S of size (k − 1) in G_v
    if found then
        return S ∪ {v}
    else
        return Empty set
```

تعداد کل مراحل بازگشتی برابر $O(2^k)$ می‌باشد، و پیچیدگی زمانی کار انجام شده در هر مرحله نیز برابر $O(m)$ است. بنابراین زمان اجرای کل الگوریتم ارائه شده از پیچیدگی زمانی $O(2^k m)$ می‌باشد.

۴ مساله‌ی فروشنده‌ی دوره‌گرد

مساله‌ی فروشنده‌ی دوره‌گرد [۸] را به شکل زیر تعریف می‌کنیم:

ورودی:
یک گراف کامل که جهت‌دار نبوده و وزن یالهای آن را اعداد بزرگ‌تر از صفر تشکیل داده‌اند.

خروجی:
یک تور با هزینه‌ی کمینه

۱.۴ ساختار زیرمسئله

برای هر راس مقصد مانند $j \in \{1, 2, ..., n\}$ و هر زیرمجموعه‌ی مانند $S \subseteq \{1, 2, ..., n\}$ که ۱ و j عضو آن هستند، در نظر می‌گیریم:

[۸] The Traveling Salesman Problem

$L_{S,j}$ طول کوتاه‌ترین مسیر از راس ۱ به j که از تمام راس‌های مجموعه‌ی S دقیقا یک‌بار می‌گذرد.

۲.۴ لم زیرمسئله‌ی بهینه

لم ۲ *فرض کنید که P کوتاه‌ترین مسیر از ۱ به j باشد که از همه‌ی راس‌های مجموعه‌ی S دقیقا یک‌بار می‌گذرد. فرض کنید یال k و j آخرین یال مسیر P باشد. در این صورت مسیر P' از ۱ به k کوتاه‌ترین مسیری است که از ۱ به k می‌رود و از همه‌ی اعضای مجموعه‌ی $S - \{j\}$ دقیقا یک‌بار می‌گذرد.*

$L_{S,j} = min\{L_{S-\{j\},k} + C_{k,j}\}, k \in S$

۳.۴ الگوریتم تحت برنامه‌ریزی پویا

Let A be a 2-D array, indexed by subsets $S \subseteq \{1, 2, ...n\}$
that contain 1 and destinations $j \in \{1, 2, ..., n\}$
Base case: $A[S, 1] = 0$ if $S = 1$ otherwise $A[S, 1] = +\infty$

Algorithm 2 Algorithm: TSP

```
function TSP
    for m = 2, 3, 4, ..., n do
        for each set S ⊆ {1, 2, ..., n} of size m that contains 1 do
            for each j ∈ S, j ≠ 1 do
                A[S_1j] = min{A[S − {j}, k] + C_kj}, k ∈ S and k ≠ j
    return min{A[{1, 2, 3, ..., n}, j] + C_j} for j = 2 to n
```

تعداد انتخاب‌ها برای j برابر n و برای S برابر ۲^n می‌باشند. همچنین زمان لازم برای پردازش هر زیر مسئله از پیچیدگی زمانی $O(n)$ تبعیت می‌کند. در نتیجه، زمان اجرای کل الگوریتم ارائه شده، از پیچیدگی زمانی $O(n^۲ ۲^n)$ خواهد بود.