دانشگاه فردوسی مشهد

دانشکده علوم ریاضی

الگوریتم پیشرفته

جستجوی کامل (Exhaustive Search)

دکتر حامد فهیمی

روش‌های Brute-Force

---

۱. مقدمه: جستجوی کامل چیست؟

بسیاری از مسائل مهم در علوم کامپیوتر نیازمند یافتن یک عنصر با ویژگی‌های خاص در دامنه‌ای هستند که اندازه آن با رشد ورودی به صورت **نمایی (Exponential)** رشد می‌کند. این مسائل معمولاً شامل اشیاء **ترکیبیاتی (Combinatorial)** مانند جایگشت‌ها (Permutations)، ترکیب‌ها (Combinations) و زیرمجموعه‌ها (Subsets) هستند.

تعریف: جستجوی کامل (Exhaustive Search) در واقع همان رویکرد Brute-force برای مسائل ترکیبیاتی است. این روش پیشنهاد می‌کند که:

1. تک‌تک عناصر دامنه مسئله (فضای حالت) تولید شوند.
2. عناصری که قیود مسئله را ارضا می‌کنند انتخاب شوند.
3. عنصر مطلوب (مثلاً عنصری که تابع هدف را ماکزیمم یا مینیمم می‌کند) پیدا شود.

اگرچه ایده این روش ساده است، اما پیاده‌سازی آن نیازمند الگوریتمی برای تولید اشیاء ترکیبیاتی (مانند تولید تمام جایگشت‌ها) است.

---

۲. بررسی مسائل کلاسیک

در این بخش سه مسئله معروف را که با روش جستجوی کامل قابل حل هستند، بررسی می‌کنیم.

الف) مسئله فروشنده دوره‌گرد (Traveling Salesman Problem - TSP)

صورت مسئله: یافتن کوتاه‌ترین توری که از مجموعه‌ای از n شهر عبور کند، به طوری که هر شهر دقیقاً یک بار بازدید شود و در نهایت به شهر مبدأ بازگردد.

- **مدل‌سازی:** این مسئله با یک گراف وزن‌دار مدل می‌شود و هدف یافتن کوتاه‌ترین **دور همیلتونی (Hamiltonian circuit)** است.

الگوریتم جستجوی کامل:
از آنجا که هر دور یک جایگشت از شهرهاست و جهت دور یا نقطه شروع تأثیری در طول دور ندارد، می‌توانیم:

1. یک شهر را ثابت در نظر بگیریم.
2. تمام جایگشت‌های $n-1$ شهر باقی‌مانده را تولید کنیم.
3. طول تور را برای هر جایگشت محاسبه کرده و کمترین را انتخاب کنیم.

تحلیل پیچیدگی:
تعداد تورهایی که باید بررسی شوند برابر است با $\frac{1}{2}(n-1)!$. این تعداد حتی برای مقادیر کوچک n بسیار بزرگ است و الگوریتم را در عمل غیرکاربردی می‌کند.

ب) مسئله کوله‌پشتی (Knapsack Problem)

صورت مسئله: n کالا با وزن‌های $w_1, \dots, w_n$ و ارزش‌های $v_1, \dots, v_n$ داریم. یک کوله‌پشتی با ظرفیت W موجود است. هدف انتخاب زیرمجموعه‌ای از کالاهاست که در کوله جا شوند و بیشترین ارزش ممکن را داشته باشند.

الگوریتم جستجوی کامل:

1. تولید تمام **زیرمجموعه‌های** ممکن از n کالا.
2. برای هر زیرمجموعه، محاسبه وزن کل و بررسی اینکه آیا از W کمتر است یا خیر (Feasibility).
3. انتخاب زیرمجموعه‌ای که بیشترین ارزش را در میان زیرمجموعه‌های مجاز دارد.

تحلیل پیچیدگی:
تعداد زیرمجموعه‌های یک مجموعه n عضوی برابر با 2^n است. بنابراین پیچیدگی این روش $\Omega(2^n)$ خواهد بود.

- **نکته:** هر دو مسئله TSP و کوله‌پشتی جزو مسائل **NP-hard** هستند و الگوریتم چندجمله‌ای شناخته‌شده‌ای برای حل دقیق آن‌ها وجود ندارد.

ج) مسئله تخصیص (Assignment Problem)

صورت مسئله: n نفر و n شغل داریم. هزینه تخصیص نفر i به شغل j برابر با $C[i, j]$ است. می‌خواهیم یک تخصیص یک‌به‌یک (هر نفر دقیقاً یک شغل) انجام دهیم تا هزینه کل مینیمم شود.

چرا روش حریصانه کار نمی‌کند؟
نمی‌توانیم صرفاً کوچکترین عدد هر سطر را انتخاب کنیم، زیرا ممکن است دو نفر کاندیدای یک شغل باشند.

الگوریتم جستجوی کامل:
هر تخصیص مجاز متناظر با یک جایگشت از اعداد 1 تا n است (تخصیص نفر i به شغل j_i).

1. تولید تمام جایگشت‌های ستون‌ها ($n!$).
2. محاسبه هزینه برای هر جایگشت.
3. انتخاب کمترین هزینه.

نکته مهم: برخلاف TSP و کوله‌پشتی، برای مسئله تخصیص الگوریتم کارآمدی به نام **روش مجاری (Hungarian Method)** وجود دارد، بنابراین استفاده از جستجوی کامل برای این مسئله توجیهی ندارد.

---

۳. حل تمرین‌های منتخب

تمرین ۱ (صفحه ۱۴۶): تحلیل کارایی مسئله کوله‌پشتی

سوال:
الف) با فرض اینکه تولید هر زیرمجموعه و بررسی وزن آن زمان ثابتی ببرد، مرتبه کارایی الگوریتم جستجوی کامل چقدر است؟
ب) اگر کامپیوتر بتواند ۱۰ میلیارد (10^{10}) محاسبات در ثانیه انجام دهد، ماکزیمم تعداد کالا (n) که می‌توان در ۱ ساعت حل کرد چقدر است؟

پاسخ:
الف) تعداد کل زیرمجموعه‌ها برای n کالا برابر با 2^n است. بنابراین کلاس کارایی الگوریتم برابر است با:

$$O(2^n)$$

ب)

- تعداد عملیات در یک ساعت:

$$\text{Ops} = 3600 \times 10^{10} = 3.6 \times 10^{13}$$

- ما باید n را چنان پیدا کنیم که $2^n \approx 3.6 \times 10^{13}$.
- با گرفتن لگاریتم در مبنای ۲:

$$n \approx \log_2(3.6 \times 10^{13})$$

$$n \approx \log_2(3.6) + 13 \times \log_2(10)$$

$$n \approx 1.85 + 13(3.32) \approx 1.85 + 43.16 \approx 45$$

نتیجه: با این ابرکامپیوتر تنها می‌توان مسئله‌ای با حدود **۴۵ کالا** را در یک ساعت حل کرد. این نشان‌دهنده محدودیت شدید الگوریتم‌های نمایی است.

تمرین ۹ (صفحه ۱۴۷): مسئله هشت وزیر (Queens-8)

سوال: در مسئله قرار دادن ۸ وزیر در صفحه شطرنج 8×8، اندازه فضای جستجو در حالات زیر چقدر است؟
الف) هیچ دو وزیری در یک خانه نباشند.
ب) هیچ دو وزیری در یک سطر نباشند.
ج) هیچ دو وزیری در یک سطر یا ستون نباشند.

پاسخ:
این تمرین نشان می‌دهد چطور فرمول‌بندی مسئله می‌تواند اندازه فضای جستجو را تغییر دهد.

الف) هیچ دو وزیری در یک خانه نباشند:
ما باید ۸ خانه را از ۶۴ خانه انتخاب کنیم.

$$\binom{64}{8} = \frac{64!}{8!56!} \approx 4.4 \times 10^9$$

تعداد حالات حدود ۴.۴ میلیارد است.

ب) هیچ دو وزیری در یک سطر نباشند:
در این حالت فرض می‌کنیم در هر سطر دقیقاً یک وزیر قرار می‌دهیم. وزیر سطر اول ۸ انتخاب دارد، سطر دوم ۸ انتخاب و ... .

$$8^8 \approx 16.7 \times 10^6$$

تعداد حالات حدود ۱۶ میلیون است. می‌بینید که با یک قید ساده، فضای جستجو بسیار کوچکتر شد.

ج) هیچ دو وزیری در یک سطر یا ستون نباشند:
در این حالت مسئله تبدیل به یافتن یک جایگشت از ستون‌ها می‌شود (مانند مسئله تخصیص). وزیر سطر اول ۸ انتخاب، وزیر سطر دوم ۷ انتخاب و ... .

$$8! = 40,320$$

تعداد حالات تنها ۴۰ هزار است. جستجوی کامل در این فضا بسیار سریع‌تر از حالت اول است.