دانشگاه فردوسی مشهد

دانشکده علوم ریاضی

الگوریتم پیشرفته

حل مسائل هندسی با روش تقسیم و غلبه (D&C)

دکتر حامد فهیمی

مسئله نزدیک‌ترین زوج نقاط و مسئله پوسته محدب

مقدمه

در مباحث قبلی دیدیم که روش‌های "الگوریتم‌های brute-force" برای مسائل هندسی معمولاً کارایی پایینی دارند. برای مثال، پیدا کردن نزدیک‌ترین زوج نقاط در صفحه با روش brute-force نیاز به زمان $\Theta(n^2)$ دارد. در این جلسه، با استفاده از تکنیک **تقسیم و غلبه**، الگوریتم‌هایی با کارایی مجانبی بسیار بهتر برای دو مسئله کلاسیک هندسه محاسباتی ارائه می‌دهیم:

1. **مسئله نزدیک‌ترین زوج (The Closest-Pair Problem)**
2. **مسئله پوسته محدب (The Convex-Hull Problem)**

---

۱. مسئله نزدیک‌ترین زوج (The Closest-Pair Problem)

تعریف مسئله

مجموعه‌ای از n نقطه $(1 < n)$ در صفحه مختصات دکارتی داریم. هدف پیدا کردن دو نقطه‌ای است که کمترین فاصله اقلیدسی را نسبت به یکدیگر دارند.

پیش‌پردازش (Preprocessing)

برای ساده‌سازی، فرض می‌کنیم نقاط متمایز هستند. قبل از شروع الگوریتم اصلی، نقاط را مرتب می‌کنیم:

- آرایه P: نقاط مرتب شده بر اساس مختصات x.
- آرایه Q: نقاط مرتب شده بر اساس مختصات y.

مراحل الگوریتم تقسیم و غلبه

اگر تعداد نقاط کم باشد $(2 \leq n \leq 3)$، مسئله به سادگی با روش brute-force حل می‌شود. اما برای $n > 3$:

1. **تقسیم (Divide):** نقاط را با یک خط عمودی $x = m$ (که m میانه مختصات x نقاط است) به دو زیرمجموعه P_l (چپ) و P_r (راست) تقسیم می‌کنیم. هر بخش تقریباً $n/2$ نقطه دارد.
2. **غلبه (Conquer):** الگوریتم را به صورت بازگشتی برای P_l و P_r فراخوانی می‌کنیم تا کمترین فاصله‌ها در سمت چپ (d_l) و سمت راست (d_r) به دست آیند.
   - فرض کنید $d = \min\{d_l, d_r\}$ باشد.
3. **ترکیب (Combine):**
   - آیا d پاسخ نهایی است؟ خیر. ممکن است نزدیک‌ترین زوج شامل یک نقطه در چپ و یک نقطه در راست باشد که فاصله آن‌ها کمتر از d است.
   - **ناحیه نواری (The Strip):** ما فقط نیاز داریم نقاطی را بررسی کنیم که در فاصله d از خط جداکننده $x = m$ قرار دارند (یک نوار عمودی با عرض $2d$).

بررسی نقاط داخل نوار (The Strip Analysis)

- لیست S را از نقاطی درون Q که در نوار قرار دارند تشکیل می‌دهیم (این نقاط قبلاً بر اساس y مرتب شده‌اند).
- برای هر نقطه p در S، تنها نیاز است نقاطی را بررسی کنیم که مختصات y آن‌ها حداکثر به اندازه d با p فاصله دارد.
- **نکته کلیدی:** از نظر هندسی می‌توان ثابت کرد که در مستطیل مورد نظر $(2d \times d)$، تعداد نقاطی که کاندیدای بررسی هستند بسیار محدود است.

- بنابراین، حلقه داخلی الگوریتم به جای چک کردن تمام نقاط، برای هر نقطه فقط تعداد محدودی (مثلاً ۵ تا ۷ نقطه بعدی) را چک می‌کند که این عمل زمان خطی $O(n)$ می‌برد.

شبه‌کد (خلاصه)

```
Algorithm EfficientClosestPair(P, Q):
1. اگر n <= 3: حل با brute-force.
2. P_l, Q_l = نیمه چپ نقاط؛ P_r, Q_r = نیمه راست نقاط.
3. d_l = EfficientClosestPair(P_l, Q_l)
4. d_r = EfficientClosestPair(P_r, Q_r)
5. d = min(d_l, d_r)
6. m = خط جداکننده
7. S = نقاطی از Q که |x - m| < d هستند.
8. برای هر نقطه i در S:
     برای k از i+1 تا i+7:
       اگر فاصله عمودی < d:
         فاصله اقلیدسی را محاسبه و d را در صورت لزوم آپدیت کن.
9. بازگرداندن d.
```

تحلیل پیچیدگی زمانی

از آنجا که مرحله تقسیم و مرحله ترکیب هر دو در زمان خطی $O(n)$ انجام می‌شوند، رابطه بازگشتی به صورت زیر است:

$$T(n) = 2T(n/2) + \Theta(n)$$

طبق قضیه اصلی (Master Theorem)، پیچیدگی زمانی برابر است با:

$$T(n) \in \Theta(n \log n)$$

مطالعه پوسته محدب اختیاری

۲. مسئله پوسته محدب (Convex-Hull Problem) - الگوریتم Quickhull

تعریف مسئله

یافتن کوچکترین چندضلعی محدب که شامل تمام n نقطه داده شده باشد.

ایده کلی (Quickhull)

این الگوریتم شباهت زیادی به Quicksort دارد.

1. نقطه‌ای با کمترین x (p_1) و بیشترین x (p_n) قطعاً رئوس پوسته محدب هستند.
2. خط واصل p_1 به p_n مجموعه نقاط را به دو بخش "پوسته بالایی" (Upper Hull) و "پوسته پایینی" (Lower Hull) تقسیم می‌کند.

مراحل ساخت پوسته بالایی

1. نقطه‌ای مانند p_{max} را پیدا کنید که بیشترین فاصله را از خط p_1p_n دارد. این نقطه حتماً یک رأس پوسته است.
2. نقاطی که درون مثلث $p_1p_{max}p_n$ قرار دارند، نمی‌توانند جزو پوسته باشند و حذف می‌شوند.
3. سپس الگوریتم به صورت بازگشتی برای دو ناحیه باقیمانده (سمت چپ خط p_1p_{max} و سمت چپ خط $p_{max}p_n$) اجرا می‌شود.

نکات پیاده‌سازی هندسی

برای تشخیص اینکه یک نقطه سمت چپ یک خط است یا برای محاسبه مساحت مثلث (جهت یافتن دورترین نقطه)، از دترمینان استفاده می‌کنیم. مساحت مثلث با رئوس $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ برابر است با نصف مقدار دترمینان زیر:

$$\text{Area} = \frac{1}{2}(x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3)$$

علامت این عبارت مشخص می‌کند نقطه q_3 در کدام سمت خط جهت‌دار q_1q_2 قرار دارد.

تحلیل پیچیدگی زمانی

- **بدترین حالت:** مشابه Quicksort، اگر تقسیم‌بندی نامتوازن باشد (مثلاً هر بار فقط یک نقطه حذف شود)، زمان اجرا $\Theta(n^2)$ خواهد بود.
- **حالت میانگین:** اگر نقاط به صورت تصادفی توزیع شده باشند، بسیاری از نقاط در هر مرحله حذف می‌شوند و زمان اجرا خطی $O(n)$ خواهد بود.

---

۳. حل تمرین‌های منتخب

تمرین ۱: مسئله نزدیک‌ترین زوج در حالت یک‌بعدی

سوال: الگوریتمی بر اساس تقسیم و غلبه برای یافتن نزدیک‌ترین دو عدد در یک مجموعه n تایی از اعداد حقیقی طراحی کنید.

پاسخ:
در حالت یک‌بعدی، نقاط روی محور اعداد (x) هستند.

1. **راه حل ساده:**
- ابتدا اعداد را مرتب می‌کنیم (Sort). این کار $O(n \log n)$ زمان می‌برد.
- پس از مرتب‌سازی، نزدیک‌ترین زوج الزاماً دو عدد مجاور در آرایه مرتب شده هستند.
- با یک پیمایش خطی $O(n)$ روی آرایه مرتب شده، تفاضل هر دو عنصر همسایه را محاسبه و کمترین را پیدا می‌کنیم.

2. **راه حل بازگشتی (مشابه ساختار دوبعدی):**
- میانه m را پیدا می‌کنیم. مجموعه را به S_L (اعداد کوچکتر از m) و S_R (اعداد بزرگتر از m) تقسیم می‌کنیم.
- به صورت بازگشتی کمترین فاصله در چپ (d_L) و راست (d_R) را می‌یابیم.
- $d = \min(d_L, d_R)$.
- در مرحله ترکیب، باید فاصله بین "بزرگترین عدد در S_L" و "کوچکترین عدد در S_R" را چک کنیم. اگر فاصله آن‌ها کمتر از d بود، جواب جدید ما آن فاصله است.

تمرین ۸: بدترین حالت الگوریتم Quickhull

سوال: بدترین کارایی زمانی Quickhull چقدر است؟

پاسخ:
بدترین حالت زمانی رخ می‌دهد که در هر مرحله از بازگشت، افراز نقاط بسیار نامتوازن باشد.

- این اتفاق زمانی می‌افتد که تمام نقاط داده شده واقعاً روی محیط پوسته محدب باشند (مثلاً نقاط روی محیط یک دایره قرار داشته باشند).
- در این حالت، هیچ نقطه‌ای در داخل مثلث حذف نمی‌شود (مرحله حذفی وجود ندارد).
- در هر مرحله بازگشتی، ما یک نقطه (p_{max}) را به عنوان رأس پیدا می‌کنیم و مسئله به دو زیرمسئله تقسیم می‌شود که یکی از آن‌ها ممکن است تهی و دیگری شامل $n-1$ نقطه باشد.
- رابطه بازگشتی مشابه بدترین حالت Quicksort می‌شود:

$$T(n) = T(n-1) + O(n) \rightarrow T(n) \in \Theta(n^2)$$

بنابراین در بدترین حالت، کارایی آن مربعی است.
