# ساختمان داده‌ها و الگوریتم‌ها

**نیم‌سال دوم ۹۷ــ۹۸**

گردآورندگان: علیرضا اکبری، پدرام خورسندی

دانشکده‌ی مهندسی کامپیوتر

پاسخ تمرین چهارم — مرتب‌سازی هرمی و سریع، درخت تصمیم

## هرم و مرتب‌سازی هرمی

### مسئله‌ی ۱. تعداد هرم های بیشینه

یک رابطه بازگشتی برای تعداد هرم های کمینه با $2^n - 1$ عنصر متمایز بیابید.

**حل.** $T(n) = {}^{2^n-2}C_{(2^n-2)/2} * T^2(n-1)$ ▷

### مسئله‌ی ۲. عنوان سوال

یک هرم بیشینه با اعداد ۱ تا ۱۰۲۴ در نظر بگیرید. چه تعداد از اعداد بزرگتر از ۱۰۰۰ میتوانند در برگ های هرم قرار گیرند؟

**حل.** حداکثر ۸ عدد بزرگتر از ۱۰۰۰ در برگ‌ها قرار میگیرد. ▷

### مسئله‌ی ۳. ادغام لیست های مرتب شده

تعداد d دنباله از عناصر ورودی داریم ، به گونه ای که هر دنباله از پیش مرتب شده و تعداد کل عناصر n تاست. الگوریتمی از $O(nlg(d))$ ارائه دهید که یک ارایه مرتب شده از همه عناصر در خروجی قرار دهد.

**حل.** عناصر ابتدای هر لیست را در یک هرم کمینه درج میکنیم. در مرحله بعد عنصر کمینه هرم را استخراج میکنیم و عنصر دوم لیست عنصر خارج شده را در هرم درج میکنیم. عنصر استخراج شده را به انتهای لیست مرتب شده اضافه میکنیم این عمل را تا خالی شدن همه لیست ها تکرار میکنیم. ▷

### مسئله‌ی ۴. آرایه نیمه مرتب

یک آرایه در نظر بگیرید که هر عنصر آن با موقعیت خود در آرایه سورت شده حداکثر به اندازه k فاصله دارد. الگوریتمی از $O(k + (n-k)logk)$ طرح کنید که این آرایه را مرتب کند.

**حل.** ابتدا از k عنصر ابتدایی آرایه یک هرم کمینه میسازیم. سپس در هر مرحله عنصر کمینه را از هرم حذف و عنصر بعدی آرایه اصلی را در هرم درج میکنیم. ▷

## مسئله‌ی ۵. یافتن میانه

یک داده ساختار طراحی کنید که هر یک از عملیات های زیر را در پیچیدگی زمانی $O(logn)$ انجام دهد.

- درج یک عنصر در داده ساختار
- دادن عنصر میانه
- حذف یک عنصر

**حل.** این داده ساختار از یک هرم کمینه و یک هرم بیشینه تشکیل میشود. لازم است تعداد اعضای هرم کمینه برابر یا حداکثر یک عضو بیشتر از هرم بیشینه باشد. در این داده ساختار ،در صورت تساوی تعداد اعضای دو هرم ،میانه میانگین ریشه های دو هرم و در صورتی که هرم کمینه یک عضو بیشتر داشته باشد ، میانه ریشه هرم کمینه خواهد بود. ▷

## مسئله‌ی ۶. چندمین کوچکترین

الگوریتمی ارائه دهید که امینk عنصر کوچک یک آرایه را در پیچیدگی زمانی مطلوب مشخص کند.

- $O(nlogn)$
- $O(n + klogn)$
- $O(n + k^۲)$

**حل.**

- داده ها را سورت میکنیم. عنصر امk آرایه سورت شده عنصر مطلوب است.
- ابتدا در $O(n)$ هرم کمینه متناظر آرایه را میسازیم. سپس k بار عنصر کمینه را استخراج میکنیم.
- عنصر امk این آرایه حداکثر در عمق k از هرم کمینه ساخته شده قرار دارد. در نتیجه عملیات حذف عنصر کمینه را برای هرمی با ارتفاع k ،متناظر با هرم ساخته شده در بخش قبل تکرار میکنیم.

▷

# درخت تصمیم

## مسئله‌ی ۷. چاه‌های آلوده

در شهری n چاه متمایز وجود دارد و مطمئن هستیم k تای آن‌ها آلوده به ویروس است، اما نمیدانیم کدامیک آلوده و کدام تمییز است. میخواهیم با حداقل تعداد آزمایش چاه‌های آلوده را تشخیص دهیم. در هر آزمایش آب‌های مجموعه را با هم قاطی و آزمایش میکنیم که آیا این آب حاوی ویروس است یا خیر.

- **الف**
بر اساس درخت تصمیمی که میسازید کران پایین تعداد آزمایش ها را به دست آورید.
- **ب**
یک الگوریتم با $O(klgn)$ برای حل مسائل ارایه دهید.

**حل.**

- **الف**
با توجه به تعداد حالات ممکن برای برگ‌ها این عملیات از $O(log(^nC_k)$ خواهد بود.
- **ب**
از search binary استفاده میکنیم. به این شکل که در ابتدا آب تعداد نصف چاه‌ها را آزمایش میکنیم. در صورت آلوده بودن الگوریتم مشابه را برای این چاه‌ها تکرار میکنیم ؛در غیر این صورت الگوریتم را برای نیمه دیگر تکرار میکنیم. در این روش برای پیدا کردن یک چاه آلوده به $O(logn)$ آزمایش نیاز داریم ؛درنتیجه، یافتن k چاه آلوده $O(klogn)$ هزینه خواهد داشت.

$\triangleright$

# مرتب‌سازی سریع و مرتب‌سازی سریع تصادفی

## مسئله‌ی ۸. مرتب‌سازی سریع، لیست پیوندی و دیگران

در الگوریتم مرتب‌سازی سریع ، به جای آرایه‌ای از اعداد ، به شما یک لیست پیوندی داده شده است بدین صورت که هر عدد تنها به عدد بعدی خود اشاره می‌کند. الگورتیم مرتب‌سازی سریع را برای این لیست پیوندی بازنویسی کنید.

**حل.**

عنصر آخر لیست پیوندی را پیدا کنید و آنرا به عنوان محور انتخاب کنید. حال عملیات partition را بدین گونه بازنویسی کنید که عنصری که از محور بزرگتر بود ، از موقعیت خود حذف شود(اشاره‌گرهای مربوط به آن می‌بایست اصلاح شود) و بعد از عنصر آخر قرار گیرد. بعد از بخش‌بندی نیز عملیات بازگشتی را روی عنصر اول(عنصر آخر لیست پیوندی مربوط به آن باید کنترل شود) و عنصری که محور به آن اشاره می‌کند اجرا کنید.

$\triangleright$

## مسئله‌ی ۹. چقدر سریع؟!

- **الف**
در یک الگوریتم مرتب‌سازی سریع تصادفی، احتمال آنکه این الگوریتم n عنصر را در زمان $\Omega(n^۲)$ مرتب کند چقدر است؟

- **ب**
در یک الگوریتم مرتب‌سازی سریع ، اگر در این الگوریتم، محور همواره عنصر میانه باشد، در آن صورت پیچیدگی الگوریتم چگونه خواهد بود؟

**حل.**

- **الف**
زمانی چنین حالتی رخ می‌دهد که الگوریتم تصادفی، بزرگترین(یا کوچکترین) عنصر را به عنوان محورانتخاب کند. احتمال آن چقدر است ؟ همچنین واضح است که هر مرحله از مرحله دیگر مستقل است.

- **ب**
در این صورت رابطه بازگشتی آن به صورت $T(n) = ۲T(\frac{n}{۲}() + O(n)$ می‌باشد. دقت کنید که به دست آوردن عنصر میانه در زمان $O(n)$ امکان‌پذیر است.(چرا؟)

$\triangleright$

## مسئله‌ی ۱۰. محور کجاست ؟ مسئله این است!

در گونه جدیدی از مرتب‌سازی سریع، برای انتخاب محور از میان n عنصر ، $۲\sqrt{n}+۱$ عنصر اول آرایه را انتخاب می‌کنیم و با یک الگوریتم ساده مانند مرتب‌سازی درجی، آن‌ها را مرتب می‌کنیم. محور، عنصر میانه این تعداد عنصر مرتب‌شده است. بقیه الگوریتم مانند قبل عمل می‌کند. بدترین زمان اجرای این الگوریتم را محاسبه کنید.

**حل.**

با این بخش‌بندی می‌توان نشان داد که هر بخش حداقل $\sqrt{n}$ عنصر دارد. حال رابطه بازگشتی آنرا بنویسید. $T(n) = T(\sqrt{n}) + T(n - \sqrt{n}) + n$ . حال این رابطه را با روش درختی تحلیل کنید. نشان دهید مجموع پیچیدگی در هر سطح از درخت از $\theta(n)$ است و همچنین با روش استقرایی نشان دهید ارتفاع درخت ((h(n)) از رابطه $h(n) \in \theta(\sqrt{n})$ پیروی می‌کند.

$\triangleright$

## مسئله‌ی ۱۱. Typical

اگر در الگوریتم مرتب‌سازی سریع تصادفی ، تمام اعداد آرایه مورد نظر با یکدیگر برابر باشند ، زمان اجرای الگوریتم را تحلیل کنید. در الگوریتم مرتب‌سازی سریع چگونه است ؟ (آیا نحوه پیاده‌سازی الگوریتم، تاثیری در محاسبه پیچیدگی دارد؟)

**حل.**

حل این مسئله به نحوه پیاده‌سازی الگوریتم مرتب‌سازی سریع وابسته است(آرایه را به ۲ بخش کوچکتر و بزرگترـ مساوی یا ۳ بخش کوچکتر، مساوی، بزرگتر از محور تقسیم کنیم.) در حالت اول برای هر دو الگوریتم برابر $O(n^2)$ است چرا که آرایه به ۲ بخش $n - 1$ عضوی و تک عضوی تقسیم می‌شود. در نحوه پیاده‌سازی دیگر پیچیدگی زمانی چگونه است ؟

$\triangleright$

## مسئله‌ی ۱۲. The Curious Case of Quick Sort

در یک الگوریتم مرتب‌سازی سریع ، این الگوریتم در هر مرحله ، آرایه ورودی را با نسبت‌های $\alpha$ و $1 - \alpha$ تقسیم می‌کند که $0 < \alpha \leqslant \frac{1}{2}$
حال نشان دهید که کمترین عمق یک برگ در درخت بازگشتی الگوریتم برابر $-\frac{lgn}{lg\alpha}$ و بیشترین عمق برابر $-\frac{lgn}{lg(1-\alpha)}$ می‌باشد.

**حل.**

برای محاسبه کم‌ترین عمق ، می‌بایست در هر مرحله زیر مسئله کوچکتر را انتخاب کنیم.(چرا؟) یعنی بخشی که متناسب با $\alpha$ است. یعنی :

$$(\alpha)^h n = 1 \to h = \log_\alpha \frac{1}{n} = -\frac{lgn}{lg\alpha}$$

برای بیش‌ترین عمق نیز به همین ترتیب عمل می‌کنیم.

$\triangleright$

موفق باشید :)