# طراحی الگوریتم

## پاسخنامه تمرین سوم - الگوریتم‌های حریصانه

مهدی نوری و امیر مهدی فرزانه

### ۱. دست گرمی — ۱۰ نمره

علی که به تازگی مغازه‌ی خود را بر پا کرده است، با مشکلی جدی رو به رو شده است. او هنگامی که می‌خواهد باقی پول مشتریانش را بدهد، گاها با این مشکل مواجه می‌شود که نمی‌تواند با سکه‌های دخلش این مقدار از باقی‌مانده را بسازد. الگوریتمی طراحی کنید که اگر علی از هر سکه با ارزش‌های مشخص به میزان نامتناهی داشت، بتواند تمامی ارزش‌ها از ۱ تا $C$ را با آنها بسازد. توجه شود که در اینجا باید کمترین تعداد سکه بکار گرفته شود. اگر مجموعه ارزش سکه‌ها را $V$ در نظر بگیریم الگوریتم شما باید از مرتبه زمانی $\mathcal{O}(|V|\log(|V|))$ باشد.

**پاسخ :**

فرض کنید $v_1 \leq v_2 \leq \cdots \leq v_k$ ترتیب انواع سکه‌ها باشد. اگر $v_1 \neq 1$، آنگاه مسئله پاسخ ندارد. برای ساختن تمام مقادیر بین صفر تا $C$، باید بتوان تمام مقادیر کمتر یا مساوی $v_1 - 1$ را ساخت.

برای بدست آوردن حداقل تعداد سکه‌های مورد نیاز، الگوریتم به این صورت عمل می‌کند:

۱. ابتدا حداقل تعداد سکه با ارزش $v_1$ را برمی‌داریم، به‌طوری‌که بتوان با استفاده از آن‌ها تمام مقادیر کمتر از $v_2$ را ساخت.

۲. سپس حداقل تعداد سکه با ارزش $v_2$ را برمی‌داریم، به‌طوری‌که بتوان با استفاده از سکه‌های قبلی و آن‌ها تمام مقادیر کمتر از $v_3$ را ساخت.

۳. به طور کلی، در تکرار $i$ام، حداقل تعداد سکه با ارزش $v_i$ را برمی‌داریم، به‌طوری‌که بتوان با استفاده از سکه‌های قبلی و آن‌ها تمام مقادیر کمتر از $v_{i+1}$ را ساخت.

برای هر تکرار $i$، مقدار $q$ به‌عنوان بزرگترین عددی که تمام مقادیر کمتر یا مساوی آن با سکه‌های موجود ساخته می‌شود، تعریف می‌کنیم. سپس تعداد سکه‌های مورد نیاز برای ساخت مقادیر کمتر از $v_{i+1}$ از رابطه زیر بدست می‌آید:

$$a_i = \max\left(\left\lceil \frac{v_{i+1} - 1 - q}{v_i} \right\rceil, 0\right)$$

بعد از این مرحله، مقدار $q$ برابر خواهد بود با:

$$q = v_i \cdot a_i + q$$

پس از این مرحله، می‌توان تمام مقادیر کمتر یا مساوی $C$ را ساخت.

پیچیدگی زمانی الگوریتم به صورت $O(|V| \log |V|)$ برای مرتب‌سازی سکه‌ها و $O(|V|)$ برای محاسبه‌ی پاسخ است، که در مجموع برابر است با:

$$O(|V| \log |V|)$$

## ۲. کدینگ ضعیف ۱۰ نمره

در مرحله ای از یک پژوهش نیاز است تا چندین حرف به صورت هافمن کد شوند. جواد از همکار خود خواست تا این کار را برای او انجام دهد. وقتی همکارش نتایج را برای او فرستاد متوجه شد که حرف $G$ را کد نکرده است.

| | کد هافمن |
|---|---|
| A | ۰۰۰ |
| B | ۰۰۱ |
| C | ۱۱ |
| D | ۱۰۰ |
| G | ؟؟ |
| E | ۱۰۱ |

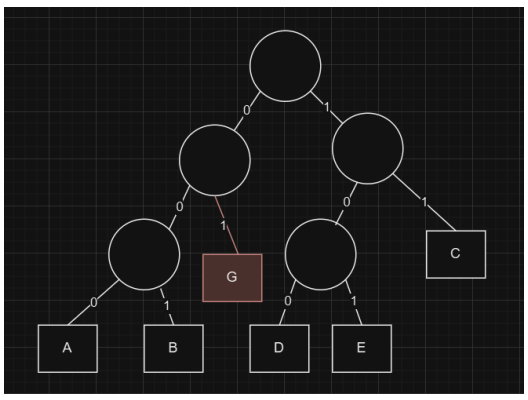
**(الف) (۲ نمره)** به او کمک کنید تا کد این حرف را بیابد.

**(ب) (۳ نمره)** با توجه به جواب قسمت قبل الگوریتمی ارائه دهید که برای $n$ حرف که یکی از آنها کد نشده در زمان خطی این کد را بیابد. $(n \geq 3)$

**(ج) (۵ نمره)** سوال قبل را برای حالتی که دو حرف بدون کد باشند، حل کنید.

**پاسخ :**

**(الف)** اگر درخت هافمن کد ها را رسم کنید می‌بینید که یک گره فقط یک برگ دارد و در نتیجه کد ۰۱ جا افتاده است.



**(ب)** در درخت هافمن رئوس میانی همگی دو فرزند دارند. در غیر این صورت می‌توان راس را با فرزندش جایگزین کرد و کد بهتری به دست آورد. اگر با $n-1$ کد داده شده درخت باینری را رسم نماییم کافی است راس میانی را که یک فرزند بیشتر ندارد بیابیم و یک فرزند به آن اضافه کنیم. کد اختصاص داده شده به راس جدید همان کد حذف شده خواهد بود.

الگوریتم:

۱) ایجاد درخت باینری با $n-1$ کد داده شده. هر راس داخلی کد مربوط به آن راس را نگه می‌دارد.

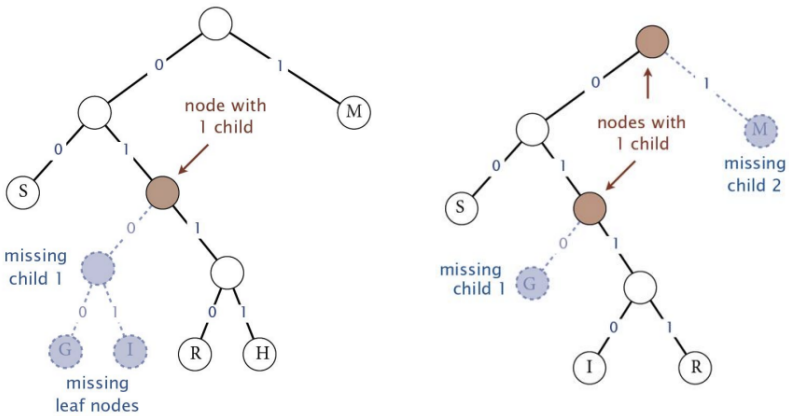
۲) پیمایش درخت و یافتن راسی داخلی که فقط یک فرزند دارد.

۳) فرزند جدید که پدر خود را به ارث می‌برد و در صورتی که فرزند سمت چپ شود باشد کد ۰ و در صورتی که فرزند سمت راست باشد ۱ به انتهای کد پدر اضافه می‌شود.

**(ج)** وقتی دو راس حذف شده باشد دو حالت ممکن است رخ دهد

۱) دو راس تک فرزند هستند. در این حالت دو راس حذف شده از فرزندان دو گره میانی مختلف می‌باشند. در این حالت یک راس مشابه قسمت ب می‌توان گراف را پیمایش کرد و گره های حذف شده را اضافه کرد.

۲) یک راس تک فرزند هستند. در این حالت دو راس حذف شده نوادگان یک راس میانی می‌باشند. در این حالت یک راس میانی باید اضافه شود و دو برگ به آن اضافه شود و دو برگ به آن اضافه شود. این حالت مشابه قسمت ب است با این تفاوت که یک راس میانی باید قبل از افزودن رئوس اضافه شود.




حالت ۲ حالت ۱

## ۳. پادشاهی تایوین ۱۵ نمره

در کشور وستروس ما $N$ شهر و $M$ جاده داریم. هر جاده برای اتصال دو شهر به هم استفاده شده است. در این کشور $K$ اژدها تارگرین داریم، که هر کدام با نام $D_i$ شناخته شده و در شهر $C_i$ زندگی می‌کند. هر اژدها در ابتدا $S_i$ سر دارد. هر اژدها به ازای هر سال زنده ماندن به اندازه $N_i$ سر به سرهایش اضافه می‌شود و تا زمانی که حداقل یک سر داشته باشد زنده می‌ماند.

حال تایوین که به دنبال تخت پادشاهی است، با تعدادی از شیرهای لنستر به نبرد اژدهایان می‌رود. این شیرها در ابتدا در یکی از شهرها قرار می‌گیرند. هر شیر در یک سال یا می‌تواند به شهر مجاور برود یا یک سر از یکی از اژدهایان شهرش را جدا کند و ببلعد. حال الگوریتمی ارائه دهید تا تایوین با کمترین شیر به نبرد اژدهایان رود و بتواند در مدت زمان متناهی تمامی این اژدهایان را نابود کند.

**پاسخ :**

ابتدا گراف این کشور را تشکیل می‌دهیم و سپس به مولفه های همبندی تقسیم می‌کنیم. واضح است که هر مولفه همبندی خود یک مسئله جداست که روی دیگر مولفه ها تاثیری ندارد پس فرض می‌کنیم گراف همبند است و مسئله را حل می‌کنیم. تعداد شیر ها را با $w$ نشان می‌دهیم. ابتدا اژدها ها را طبق $N_i$ مرتب می‌کنیم. می‌دانیم که برای کشتن اژدهای $D_i$ دو حالت وجود دارد یا اینکه $w > N_i$ باشد یا اینکه در روز نخست حداقل $S_i$ شیر به جنگ آن اژدها بروند. واضح است که اگر $w \leq N_i$ آنگاه فقط در حالتی می‌توان با $w$ شیر مسئله را حل کرد که $w \geq \sum_{j=i}^{k}(S_j)$ ( چون وقتی بر حسب $N_i$ مرتب کرده ایم جلوی رشد سر های با $N_i$ کوچکتر را می‌توانیم بگیریم ولی برای اژدها های بعدی ناچاریم به تعداد سر های اولیه آنها شیر داشته باشیم). از سوی دیگر اگر $wN_i$ می‌توان همه اژدها های $D_j : j \leq i$ را کشت. پس برای اینکه تعداد شیر ها برای هر دوی این حالات کافی باشد بین آنها ماکزیمم می‌گیریم. بنابراین جواب برابر است با :

$$\min_{i\in\{1,\dots,k\}} \left( \max \left( N_{i-1}+1, \sum_{j=i}^{k} S_j \right) \right)$$

مرتب سازی اژدها ها $O(klog(k))$ است و بقیه الگوریتم $O(k)$ زمان می‌برد.

## ۴. جاده‌های کوهستانی ۱۵ نمره

پادشاهی آفتاب در سرزمین کوهستانی حکمرانی می‌کند و تصمیم گرفته تا گنجینه‌های افسانه‌ای را که در مسیرهای پنهان میان شهرهای این سرزمین گم شده‌اند، بازپس گیرد. برای این مأموریت بزرگ، پادشاه تعداد $k$ تیم کاوشگر شجاع در اختیار دارد که هر تیم با یک درشکه جادویی و سرعتی ثابت به جستجوی مسیرها می‌پردازد.

در این مسیر کوهستانی پرپیچ‌وخم، ایستگاه‌های توقفگاهی وجود دارد که تیم‌های کاوشگر می‌توانند در آن‌ها استراحت کنند و سپس به مسیر خود ادامه دهند. فاصله بین هر دو ایستگاه مشخص است.

پادشاه می‌خواهد این مأموریت با کمترین زمان و هزینه ممکن انجام شود، به شرط آنکه قوانین زیر رعایت شوند:

- هر بخش از جاده بین دو ایستگاه باید توسط یک تیم، به طور کامل، کاوش شود.

- هر تیم باید مسیر پیوسته‌ای از جاده را کاوش کند و نمی‌تواند چند بخش غیرمتوالی را بررسی کند. به عبارت دیگر، اگر تیمی بین ایستگاه‌های ۳ تا ۴ را کاوش کرد، نمی‌تواند بدون طی کردن فاصله ۴ تا ۵، مستقیماً به ایستگاه ۵ برود.

الگوریتمی طراحی کنید که کمترین زمان لازم برای کاوش کامل مسیر جاده‌های کوهستانی را با استفاده از $k$ تیم کاوشگر محاسبه کند.

**پاسخ :**

برای حل این مسئله از الگوریتم عملیات جستجو دودویی بر روی زمان جستجو استفاده می‌کنیم. مراحل حل مسئله به این صورت است:

حد پایین $L = 0$ و حد بالا برابر با جمع کل زمان لازم برای طب مسافت‌های بین ایستگاه‌ها است.

- در هر مرحله، بررسی می‌کنیم که آیا می‌توان در زمان $t$ (یعنی نقطه میانی فعلی) با $k$ تیم کل جاده را جستجو کرد یا خیر.
- $v$ برابر سرعت خودرو ها می‌باشد
- از ابتدای جاده شروع به تخصیص تیم‌ها می‌کنیم. هر تیم تا زمانی که مجموع مسافت‌های طی شده آن از $t * v$ بیشتر نشود، بخش‌های متوالی جاده را بررسی می‌کند.
- اگر در طول مسیر به نقطه‌ای برسیم که مجموع مسافت بیشتر از $t * v$ شود، تیم جدیدی به کار گرفته می‌شود.
- اگر توانستیم تمام جاده را با $k$ تیم یا کمتر پوشش دهیم، یعنی زمان $t$ کافی است و به سراغ نیمه اول اعداد میرویم و در غیر اینصورت نیمه دوم را بررسی می کنیم

## ۵. سیستم ضعیف ACM ۲۰ نمره

می‌خواهیم سیستم مدیریت فایل انجمن علمی ACM را بهینه‌سازی کنیم تا هزینه‌ی دسترسی به فایل‌ها کمینه شود. در این سیستم، هر فایل با نام $f_i$ و اندازه‌ی $b_i$ مشخص شده است. احتمال دسترسی کاربران به هر فایل $p_i$ بوده و پس از هر بار دسترسی، فایل به موقعیت اولیه بازمی‌گردد. بنابراین، زمان دسترسی به هر فایل با فاصله‌ی موقعیت انتهایی آن از ابتدای حافظه مرتبط است.

به‌طور کلی، زمان دسترسی به یک فایل به فاصله‌اش از ابتدای حافظه و احتمال دسترسی کاربر به آن بستگی دارد. از آنجا که فایل‌ها پس از هر دسترسی به جای قبلی خود بازمی‌گردند، ترتیب ذخیره‌سازی آن‌ها تأثیری در فاصله‌ی دیگر فایل‌ها از ابتدای حافظه ندارد. در هر دسترسی، زمان لازم به فاصله از ابتدای حافظه تا فایل و همچنین احتمال دسترسی به فایل وابسته است.

هدف طراحی الگوریتمی است که هزینه‌ی دسترسی مجموع فایل‌ها را به کمترین مقدار ممکن برساند. اثبات بهینگی الگوریتم مورد نظر و همچنین تحلیل مرتبه‌ی زمانی آن نیز لازم است.

**پاسخ :**

فایل‌ها را بر حسب $\frac{b_i}{p_i}$ آن‌ها به‌صورت صعودی مرتب می‌کنیم. همانطور که واضح است هزینه‌ی اجرای این الگوریتم $O(n \log n)$ است.

فرض کنید ترتیب به‌دست‌آمده از الگوریتم ما به این صورت باشد:

$$\{\dots, f_i, \dots, f_j, \dots\}$$

که طبق الگوریتم حریصانه می‌دانیم:

$$\frac{b_i}{p_i} < \frac{b_j}{p_j}$$

چون حجم فایل‌ها و احتمال دسترسی به آن‌ها عددی غیر منفی است، پس داریم:

$$b_i \times p_j - p_i \times b_j < 0$$

در نظر می‌گیریم که جواب بهینه تا قبل از انتخاب $f_i$ با جواب حریصانه مشترک است اما در اینجا با هم تفاوت می‌کنند و به صورت زیر است:

$$\{\dots, f_j, \dots, f_i, \dots\}$$

هزینه‌ی جواب بهینه نسبت به جواب حریصانه به این صورت است که فایل $f_i$ با $b_j + m$ واحد جلو رفته که $m$ تعداد بایت مبانی است. پس هزینه را به اندازه‌ی $p_i \times (b_j + m)$ افزایش داده است. همچنین فایل $f_j$ به اندازه‌ی $p_j$ هزینه را نسبت به حالت قبل کاهش می‌دهد. در مجموع هزینه به اندازه‌ی $m \times (p_i - p_j)$ تغییر کرده است که:

$$m > 0,\ p_i > p_j \implies m(p_i - p_j) > 0$$

$$b_i \times p_j - p_i \times b_j < 0 \implies p_i \times b_j - b_i \times p_j > 0$$

پس هزینه نسبت به حالت قبل بیشتر شده و این تناقض اثبات می‌کند که الگوریتم حریصانه بهینه است.

## ۶. پروانه‌ها و گل‌ها ۲۰ نمره

علی یک نقاش حرفه‌ای است که برای تامین مایحتاج زندگی، به سفارش افراد نقاشی طراحی می‌کند و حال تصمیم دارد اثر بعدی خود را با عنوان باغ گل و پروانه نقاشی کند. طبق سفارش مشتری، در این نقاشی باید تعداد برابری گل و پروانه وجود داشته باشد، همچنین اندازه هر گل و پروانه نیز از پیش توسط مشتری تعیین شده است.

مشتری درخواست کرده هر پروانه تنها بر روی یک گل باشد و روی هر گل نیز بیش از یک پروانه نباشد.

علی برای هر گل و پروانه عددی به‌عنوان اندازه‌ی آن‌ها تعیین می‌کند و باید تلاش کند تا جمع اختلاف اندازه بین پروانه‌ها و گل‌های متناظرشان حداقل شود، تا نقاشی باغ به زیباترین حالت ممکن درآید.

**مثال:** فرض کنید دو پروانه با اندازه‌های ۱ و ۴ و دو گل با اندازه‌های ۲ و ۵ داریم. باغ در زیباترین حالت خواهد بود اگر پروانه‌ی اول روی گل اول و پروانه‌ی دوم روی گل دوم قرار گیرد.

علی راه‌حلی به ذهنش رسیده است: او پروانه‌ها و گل‌ها را به ترتیب اندازه مرتب می‌کند، سپس پروانه‌ها را به گونه‌ای نقاشی می‌کند که هر پروانه تا جای ممکن با گلی که کمترین اختلاف اندازه را دارد، جفت شود. ابتدا پروانه‌هایی که کاملاً با اندازه گل‌ها مطابقت دارند، نقاشی می‌شوند؛ سپس پروانه‌هایی با اختلاف اندازه‌ی ۱ و سپس با اختلاف ۲، و به

همین ترتیب تا تمام پروانه‌ها نقاشی شوند.

- درستی این راه‌حل را بررسی کنید. اگر درست است، آن را اثبات کنید.
- در صورت نادرست بودن، مثال نقض بیاورید و الگوریتمی صحیح ارائه دهید و درستی آن را اثبات کنید.
- در هر حالت، پیچیدگی زمانی راه‌حل خود را محاسبه کنید.

**پاسخ :**

راه حل مطرح شده، نادرست است. مثال نقض :

پروانه ها با اندازه ۴و۱۰ و گل ها با اندازه ۱و۶ می باشند.

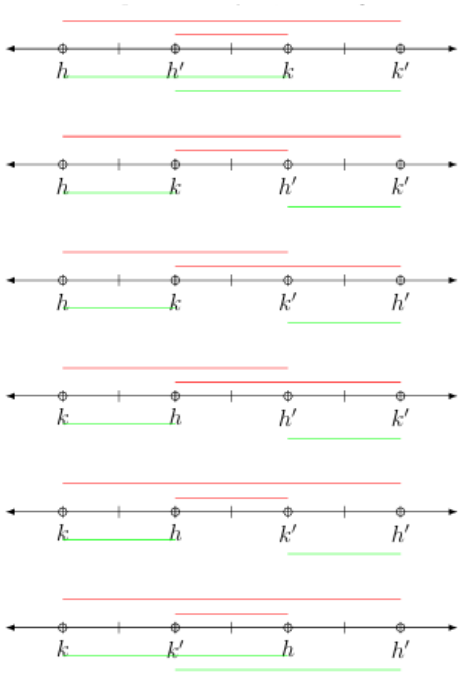
در اینصورت علی پروانه با اندازه ۴ را به گل با اندازه ۶ اختصاص می دهد و پروانه اندازه ۱۰ را بر روی گل با اندازه یک می کشد در اینصورت جمع اختلاف میشود ۱۱.

اما در روش بهینه پروانه ۴ بر روی گل ۱ و ۱۰ بر روی گل ۶ کشیده میشد که در آنصورت جمع اختلاف ، ۷ بدست می آمد.

الگوریتم صحیح:

۱. ابتدا لیست پروانه‌ها و گل‌ها را به ترتیب اندازه از کوچک به بزرگ مرتب می‌کنیم.
۲. سپس کوچک‌ترین پروانه را به کوچک‌ترین گل اختصاص می‌دهیم، دومین کوچک‌ترین پروانه را به دومین کوچک‌ترین گل اختصاص می‌دهیم، و به همین ترتیب تا انتها.

- مرتب‌سازی لیست‌ها هر کدام با زمان $O(n \log n)$ قابل انجام است.
- تخصیص پروانه‌ها به گل‌ها با زمان $O(n)$ انجام می‌شود.
- بنابراین، زمان کلی الگوریتم $O(n \log n)$ است که یک زمان چندجمله‌ای می‌باشد.




شکل ۱: اندازه پروانه ها و گل ها در حالات مختلف

اثبات درستی:

تصور کنید که الگوریتم بهینه‌ای وجود دارد که کاملاً شبیه الگوریتم مطرح شده است، یعنی به کوچکترین گل، کوچکترین پروانه اختصاص داده شده است، بجز یک جفت پروانه و گل. اندازه جفت پروانه $h$ و $h'$ می‌باشد و گل‌ها $k$ و $k'$ هستند.

در الگوریتم اولیه، $h$ به $k$ و $h'$ به $k'$ اختصاص می‌یابد اما در راه‌حل بهینه، برعکس خواهد بود.

در شکل یک، تمام حالات نشان داده شده است که در صورتی که از الگوریتم اولیه برای تخصیص این دو جفت استفاده کنیم، یا بهتر از حالت بهینه خواهد شد و یا مساوی آن خواهد بود. بنابراین، الگوریتم اولیه بهینه است.

حالت سبز رنگ مربوط به الگوریتم بهینه و قرمز رنگ الگوریتم غیر بهینه است.

## ۷. سنگ‌های فضایی ۲۰ نمره

در یک مأموریت ناسا، گروهی از فضانوردان به سیاره‌ای دوردست رسیدند و با مجموعه‌ای از سنگ‌های فضایی با وزن‌های مختلف روبه‌رو شدند. هر سنگ دارای خاصیت منحصر به فردی بود و برای انجام آزمایش‌های تحقیقاتی مختلف باید این سنگ‌ها را به $k$ گروه تقسیم می‌کردند.

به فضانوردان لیستی از وزن سنگ‌ها داده شد و آن‌ها تصمیم گرفتند سنگ‌ها را طبق قوانین زیر گروه‌بندی کنند:

- هیچ گروهی نباید خالی بماند.
- اگر سنگ $i$ام و سنگ $j$ام در لیست وزن‌ها در یک گروه باشند، تمام سنگ‌هایی که وزنشان بین این دو قرار دارد نیز باید در همان گروه باشند.

رییس تیم فضانوردان معیاری برای ارزیابی عملکرد فضانوردان طراحی کرد:

- امتیاز هر گروه برابر است با جمع امتیاز اولین و آخرین سنگ آن گروه.
- امتیاز نهایی هر آزمایش برابر است با جمع امتیازات تمام گروه‌ها.

به رییس فضانوردان کمک کنید تا اختلاف امتیاز بهترین و بدترین حالت گروه‌بندی را برای آزمایش محاسبه کند.

در نهایت، پیچیدگی زمانی راه‌حل خود را برای محاسبه این اختلاف ارزیابی کنید.

برای درک بیشتر مسئله، به مثال زیر توجه کنید.

**مثال:** فرض کنید لیست وزن سنگ‌ها به صورت $[3, 2, 5, 1]$ و تعداد گروه‌ها $k = 2$ باشد. در این حالت:

- بیشترین امتیاز زمانی حاصل می‌شود که یک گروه سنگ‌های اول و دوم را شامل شود و گروه دوم شامل سنگ سوم و چهارم باشد. در این صورت، امتیاز آزمایش برابر خواهد بود با:

$$(3 + 2) + (5 + 1) = 11$$

- کمترین امتیاز زمانی حاصل می‌شود که اولین سنگ در یک گروه و سنگ‌های دوم تا چهارم در گروه دیگری باشند. در این صورت، امتیاز نهایی برابر خواهد بود با:

$$(3 + 3) + (2 + 1) = 9$$

بنابراین، اختلاف امتیاز بین بهترین و بدترین حالت برابر است با 2.

**پاسخ :**

برای حل این مسئله، هدف ما این است که سنگ‌ها را به $k$ گروه تقسیم کنیم، به گونه‌ای که تفاوت بین بیشترین و کمترین امتیاز ممکن از این تقسیم‌بندی‌ها را بیابیم. امتیاز هر گروه از سنگ‌ها، مجموع وزن اولین و آخرین سنگ آن گروه است.

مراحل حل مسئله به این صورت است:

۱. محاسبه وزن هر جفت سنگ مجاور: برای اینکه بتوانیم ترکیب‌های مختلفی از سنگ‌ها را بررسی کنیم، ابتدا وزن هر جفت سنگ مجاور را محاسبه می‌کنیم. به این ترتیب، اگر سنگ‌های $i$ و $i+1$ مجاور باشند، مجموع وزن آن‌ها را به عنوان یک وزن جفت در نظر می‌گیریم و در یک لیست جدید ذخیره می‌کنیم. این کار به ما کمک می‌کند تا سریع‌تر به بیشترین و کمترین وزن‌ها دسترسی داشته باشیم.

۲. مرتب‌سازی وزن‌های جفت‌ها: بعد از محاسبه وزن‌های جفت سنگ‌های مجاور، آن‌ها را به ترتیب صعودی مرتب می‌کنیم. این ترتیب به ما اجازه می‌دهد تا به راحتی کمترین و بیشترین امتیازهای ممکن را برای تشکیل گروه‌ها انتخاب کنیم.

۳.محاسبه بیشترین و کمترین امتیاز برای تقسیم‌بندی سنگ‌ها: - برای محاسبه بیشترین امتیاز، ابتدا وزن اولین و آخرین سنگ کل مجموعه را در امتیاز خود لحاظ می‌کنیم و سپس $k-1$ جفت از بزرگ‌ترین وزن‌ها را به این امتیاز اضافه می‌کنیم. این کار به ما بیشترین امتیاز ممکن از تقسیم‌بندی را می‌دهد. - برای محاسبه کمترین امتیاز، مشابه بالا عمل می‌کنیم، با این تفاوت که $k-1$ جفت از کوچک‌ترین وزن‌ها را به امتیاز اضافه می‌کنیم. این کار به ما کمترین امتیاز ممکن را می‌دهد.

۴. محاسبه تفاوت امتیازها: در نهایت، تفاوت بین بیشترین و کمترین امتیازها را به عنوان نتیجه مسئله برمی‌گردانیم. این تفاوت نشان می‌دهد که چگونه انتخاب‌های مختلف برای گروه‌بندی سنگ‌ها می‌تواند امتیاز نهایی را تحت تأثیر قرار دهد.

با توجه به مرتب سازی مرحله دوم ، پیچیدگی زمانی برابر با $\mathcal{O}(nlog(n))$ خواهد بود.