راهنمای گام به گام

به پروژه صفر درس ساختمان داده‌ها و الگوریتم‌ها خوش آمدید!

در این راهنما قصد داریم قدم به قدم با هم پیش برویم و از ساده‌ترین روش، شروع به حل یک مسئله زیبا کنیم.

مطالعه‌ی راهنما قبل از کارگاه خالی از لطف نیست و سعی کرده‌ایم راهنما را طوری بنویسیم که به تنهایی و خارج از کارگاه هم بتوانید پیش بروید. در حین کارگاه هم از روی همین راهنما پیش خواهیم رفت. همچنین دستیاران آموزشی در کارگاه با شما خواهند بود تا در ابهامات و اشکالات احتمالی همراهیتان کنند.

مقدمه

همگی ما با مبانی برنامه‌نویسی *C* آشنایی داریم و احتمالاً به واسطه درس برنامه‌سازی پیشرفته با *Java* هم آشنا شده‌ایم. همچنین هیچ بعید نیست عده‌ای از شما برنامه‌نویسی به زبان *Python* را هم بلد باشید. در پروژه‌های این درس از زبان‌های *C* و یا *C++* بهره خواهیم برد. (پایتون هم به پروژه صفر اضافه گردید.)

می‌دانستیم که هر برنامه برای اجرا مقدار مشخصی زمان نیاز دارد که این زمان تقریباً با تعداد دستورهای اجرا شده در آن برنامه متناسب است. در درس «ساختمان‌داده‌ها و الگوریتم‌ها» با مفهوم نماد O و احتمالاً Ω و Θ آشنا شدیم و حالا در این پروژه می‌خواهیم دست به کد شویم و زمان مورد نیاز برای اجرای راه‌حل‌های متفاوت با اُردرهای زمانی متفاوت را به طور عملی ببینیم.

چاپ زمان اجرای کد

با یک سرچ ساده‌ی «زمان اجرا زبان فلان» یا "execution time folan language" می‌توان به سادگی مطالب زیر را پیدا کرد که ما برای راحتی شما و سرعت بخشیدن به کارگاه این کار را برایتان از قبل کرده‌ایم. با کلیک بر روی هر کدام از جعبه‌های زیر نحوه چاپ زمان اجرا را در زبان مورد نظر ببینید و حتما در سیستم خودتان هم تست کنید! خروجی این برنامه‌ها به میلی‌ثانیه است.

کد C++
کد Python

(با اجرای کد کامنت شده، و تغییر عدد ثابت آن می‌توانید حدود تعداد عملیاتی که سیستم شما در ۱ ثانیه اجرا می‌کند بدست آورید.)

صورت مسئله

دنباله‌ای از اعداد صحیح به طول nn به ما داده شده است. به هر بازه‌ی پیوسته‌ی ii تا jj از دنباله یک «**زیردنباله**» می‌گوییم. پس nn زیردنباله به طول ۱ داریم و همچنین n−1n−1 زیردنباله به طول ۲ داریم و ...

حال تمامی زیردنباله‌های ممکن را تصور کنید. خواسته‌ی مسئله، یافتن بیشینه‌ی مجموع اعداد از بین تمام زیردنباله‌هاست.

برای مثال به نمونه زیر دقت کنید:

```
[index :[0 1 2 3 4 5 6 7 8
numbers: 5 -7 10 2 -4 5 -7 6 -2
maximum sum:  ^.....^ = 10+2-4+5 = 13
```
Plain text

بررسی کنید که آیا هیچ زیردنباله دیگری مجموع بیش‌تری نداشت؟!

فرمت ورودی و خروجی مسئله

در خط اول ورودی عدد nn می‌آید که نشان‌دهنده اندازه دنباله ورودی است و در خط بعدی nn عدد صحیح با فاصله از یکدیگر می‌آیند. در خروجی کافیست بزرگ‌ترین مجموعِ زیردنباله ممکن را چاپ کنید.

ورودی نمونه

```
9
2- 6 7- 5 4- 2 10 7- 5
```
Plain text

خروجی نمونه

```
13
```
Plain text

نحوه ارزیابی پروژه

همانطور که مشاهده می‌کنید پروژه بر اساس اینکه اندازه دنباله اولیه یعنی nn تا چه اندازه بزرگ باشد به ۴ زیرمسئله تقسیم شده و همچنین در انتها یک نمودار نیز از شما خواسته شده است.

ایده و راه‌حل هر ۴ زیرمسئله را در ادامه با هم می‌بینیم توضیحات نمودار خواسته شده هم در سوال خودش توضیح داده شده است. توصیه می‌شود کدهای هر ۴ زیرمسئله‌تان را نگه‌دارید و با روش خواسته شده پیاده‌سازی کنید چرا که در رسم نمودار مجددا به آن‌ها نیاز پیدا می‌کنید.

نمره‌ی زیرمسئله‌های یکم تا چهارم توسط داوری آنلاین کوئرا داده خواهد شد که در آن اولا صحیح بودن خروجی کد شما در ازای تعدادی ورودی و ثانیا مدت زمان اجرای کد شما برای هر ورودی سنجیده می‌شود. (که اگر توضیحات زیر را دنبال کنید، احتمالا به مشکلی بر نخواهید خورد.)

نمره‌ی بخش «نمودار»، توسط دستیاران آموزشی داده خواهد شد. توضیحات بیشتر در مورد بخش «نمودار» در بخش خودش آورده شده است.

بر روی هر زیرمسئله فقط به راه‌حل‌های هم اُردر (هم تتا در واقع) با راهنمای آن نمره تعلق می‌گیرد. (یعنی ارسال راه‌حل زیرمسئله ۱ در زیرمسئله ۲ یا برعکس مجاز است، اما نمره‌ای به آن تعلق نمی‌گیرد.) این موضوع توسط دستیاران آموزشی بررسی خواهد شد. در صورت انجام این کار، هرچند داوری کوئرا به شما نمره‌ی کامل داده باشد، نمره‌ی آن زیرمسئله برای شما صفر در نظر گرفته خواهد شد. همچنین توجه داشته باشید که «ارسال نهاییِ» شما، تنها ارسال شما محسوب خواهد شد.

زیرمسئله یکم

اول از همه بیایید ساده‌ترین راه‌حل ممکن را برای مسئله پیاده‌سازی کنیم. دو متغیر ii و jj در نظر بگیرید به کمک این دو و با استفاده از حلقه‌ی تودرتو تمام شروع و پایان‌های ممکن را برای زیردنباله در نظر بگیرید. حال به ازای هر حالت از ii و jj روی تمام عناصر با اندیس‌های ii تا jj فور بزنید و مقادیر آن‌ها را با هم جمع کنید تا مجموع عناصر این زیردنباله بدست بیاید. حال این مجموع را بیشترین مجموعی که تا بحال به دست آورده‌ایم ماکسیمم بگیرید و به سراغ ii و jjهای بعدی بروید.

سعی کنید الگوریتم بالا را پیاده‌سازی و سپس تحلیل اُردر کنید!

اُردر

محاسبه زمان تقریبی اجرا

محاسبه زمان تقریبی اجرای برنامه‌ها کار دشواری نیست. عموما این موضوع به سیستمی که کد را اجرا می‌کند هم مربوط می‌شود اما یک استاندارد و حدود مشخصی دارد و در واقع می‌تواند نشان دهد که کدمان مثلا یک ساعت زمان برای اجرا نیاز ندارد و در حدود یک ثانیه یا کمتر به جواب می‌رسد.

استاندارد حدودی این‌گونه است: تعداد عملیات‌های برنامه را می‌شماریم، در یک برنامه به زبان *C* یا *C++* فرض می‌کنیم که هر $2*1082*108$ عملیات در حدود یک ثانیه اجرا می‌شود. این عدد به سخت‌افزار و قدرت پردازش سیستم هم بستگی دارد. (توی پرانتز بگم که رزرو کردن حافظه در هنگام شروع اجرای برنامه هم تا حدی زمان نیاز دارد، مثلا وقتی یک آرایه‌ی خیلی خیلی بزرگ تعریف می‌کنیم زمان اجرای برنامه هم زیاد میشه. فعلا تا وقتی به مشکلش برنخوردید این پرانتز رو نادیده بگیرید.)

در این سوال ما برنامه‌ای با حدود $n3n3$ عملیات نوشتیم. به سوال «زیرمسئله یکم» بروید و محدودیت nn و به خصوص حداکثر مقدارش را مشاهده کنید. آیا $n3n3$ عملیات در کمتر از یک ثانیه انجام می‌شود؟

خب حالا با همین برنامه به سراغ «زیرمسئله دوم» بروید. سنگ مفت، گنجشک مفت، شاید اکسپت شد! البته به محدودیت nn در این سوال هم گوشه چشمی داشته باشید. حدود $n3n3$ عملیات با این nn در یک ثانیه انجام می‌شود؟

قبل از رفتن به سراغ زیرمسئله دوم بهتر است «یکُم» را اکسپت کرده باشید. اگر در کارگاه هستید و به مشکلی خوردید با دستیارآموزشی مطرح کنید.

زیرمسئله دوم

در این زیرمسئله اندازه ورودی بزرگ‌تر خواهد بود و الگوریتم قبلی ما جوابگوی حل آن در زمان مناسب نیست. چند دقیقه‌ای فکر کنید و سعی کنید یکی از حلقه‌ها را حذف کنید...

خب حالا که فکر کردید، فرض کنید ii را فیکس کرده باشیم و در حال فور زدن روی jj باشیم. فرض کنید مجموع زیردنباله ii تا j−1j−1 را داشته‌ایم، آیا برای محاسبه مجموع زیردنباله ii تا jj فور زدن روی تمامی عناصرش نیاز است؟ کافی نیست که همان مجموع زیردنباله ii تا j−1j−1 را با عنصر jjاُم جمع کنیم؟ با توجه به این موضوع از ۳ حلقه تودرتوی ما یکی به سادگی حذف می‌شود. همچنین می‌توانید اُردر زمانی الگوریتم جدید را هم حساب کنید. حالا با این الگوریتم زیرمسئله دوم هم به سادگی اکسپت می‌شود و حدود n2n2 عملیاتش در یک ثانیه می‌گنجد.

اُردر

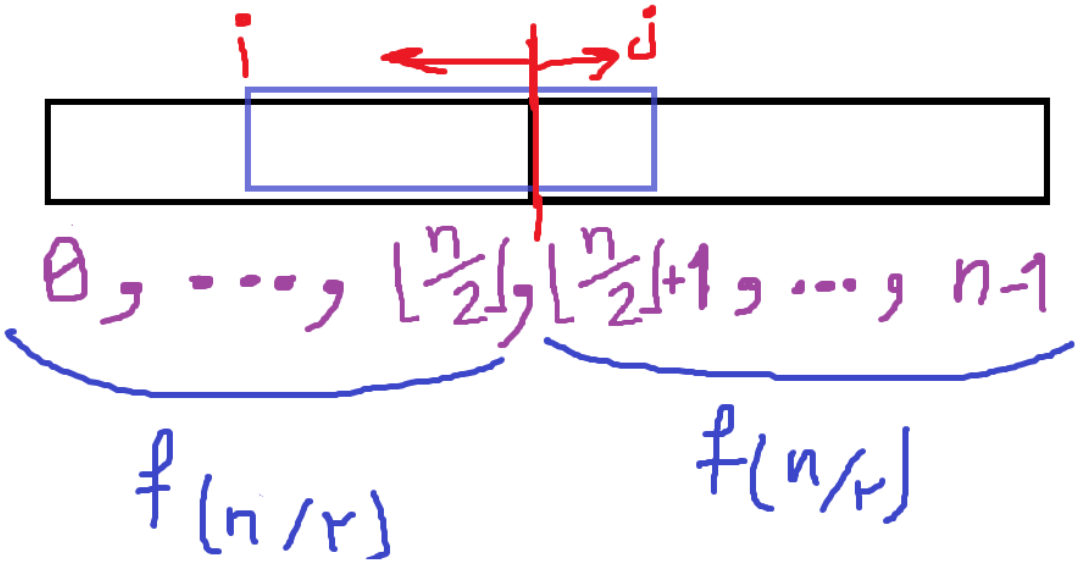
پس از گرفتن اکسپت زیرمسئله دوم، کدتان را روی زیرمسئله سوم هم می‌توانید تست کنید. البته قبل از آن بد نیست محدودیت nnش را نگاه کنید.

قبل از رفتن به سراغ زیرمسئله سوم بهتر است «دوم» را اکسپت کرده باشید. اگر در کارگاه هستید و به مشکلی خوردید با دستیارآموزشی مطرح کنید.

زیرمسئله سوم

خب همانطور که می‌دانید اگر الگوریتم قبلی را روی زیرمسئله سوم ارسال کنید، با خطای محدودیت زمانی یا *Time Limit Exceeded* مواجه می‌شوید و اکسپت نمی‌شود. باید به سراغ الگوریتمی سریع‌تر برویم. اگر دوست داشتید کمی در این باره فکر کنید و سپس ادامه متن را مطالعه کنید. کمی هم با دانستن اینکه شاید یک الگوریتم بازگشتی بتوانیم بنویسیم به مسئله فکر کنید.

خب برویم سراغ راه‌حل سوم. فرض کنید حل کردن مسئله را به ازای kkهای کوچک‌تر از nn بلد هستیم. دنباله را به دو نیمه‌ی تقریبا مساوی تقسیم می‌کنیم. هر کدام از نیمه‌ها را طبق فرض حل می‌کنیم و بیشینه مجموع زیردنباله‌هایش را بدست می‌آوریم. جواب برای دنباله اصلی ما یا یکی از این دو عدد به دست آمده است و یا مجموع عناصر زیردنباله‌ای است که بخشی از آن در نیمه سمت راست دنباله و بخشی از آن هم در نیمه سمت چپ دنباله است. (چرا که اگر کامل در یکی از این دو نیمه باشد هنگام حل آن نیمه حتما مورد بررسی قرار گرفته است.)



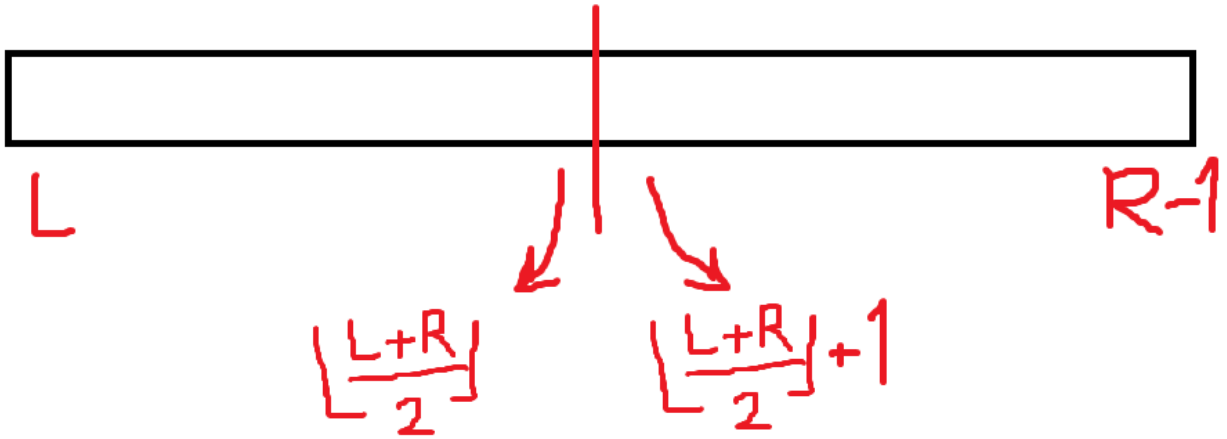
سعی کردیم برای فهم بهتر موضوع تصویری هم آماده کنیم. خب پس جواب یا در یکی از f(n/2)f(n/2) هاست و یا در زیردنباله‌ای مثل ii تا jj. خب یک نکته‌ای که اینجا داریم این است تمام زیردنباله‌هایی که حالا دنبالشان هستیم از چسباندن دو زیردنباله که یکی حتما از n2+12

n+1 شروع می‌شود و یکی حتما به n22n ختم می‌شود تشکیل می‌شود. پس کافیست یک حلقه بیش‌ترین جمع برای زیردنباله‌هایی که از n2+12n+1 شروع می‌شود را پیدا کنیم و با یک حلقه هم بیش‌ترین جمع برای زیردنباله‌هایی که به n22

n ختم می‌شوند. مجموع این دو می‌شود بیشینه مد نظر ما که باید با f(n/2)f(n/2)ها مقایسه شود و یکی از این ۳ جواب نهایی است. هنوز فقط الگوریتم را گفته‌ایم و در ادامه پیاده‌سازی را هم خواهیم گفت. اما قبل از آن بیایید تحلیل اُردر کنیم. ابتدا سعی کنید خودتان حل کنید و سپس به سراغ جعبه زیر بروید.

اُردر

خب حالا الگوریتم را می‌دانیم. چطور پیاده‌سازی کنیم؟ آرایه اعداد را به صورت گلوبال و بالای کد تعریف می‌کنیم، یک تابع بازگشتی می‌نویسیم که ورودی‌اش، اندیس شروع و پایان بازه مدنظر را می‌گیرد و مسئله را برای آن زیردنباله حل می‌کند. اکثر اوقات بازه‌ها را به صورت بسته-باز می‌گیریم. یعنی ابتدای آن‌ها را بسته و انتهای آن‌ها را باز. یا به عبارتی به این صورت: [start,end)(start,end) خب حالا به تصویر زیر نگاه کنید:



در F(L,R)F(L,R) چه کارهایی باید کنیم؟ در توابع بازگشتی ابتدا حالت پایه را می‌نویسیم. کافیست هرگاه اندازه بازه ۱ بود همان عدد array[L]array[L] را برگردانیم (البته اگر آن یک عدد منفی بود، صفر را برمیگردانیم). بعد از مشخص کردن و درست کردن پایه باید

1. نیمه اول را صدا بزنیم تا به صورت بازگشتی جوابش محاسبه شود.
2. نیمه دوم را صدا بزنیم تا به صورت بازگشتی جوابش محاسبه شود.
3. و در آخر با دو حلقه (یکی از وسط به آخر و دیگری از وسط به ابتدا) بیشینه‌ی مجموع زیردنباله‌های بنفش گونه (در تصور اولی) را به دست آوریم. بیشینه‌ی این سه، خروجی تابع ما خواهد بود.

قبل از رفتن به سراغ زیرمسئله چهارم بهتر است «سوم» را اکسپت کرده باشید. اگر در کارگاه هستید و به مشکلی خوردید با دستیارآموزشی مطرح کنید.

زیرمسئله چهارم!

شاید تعجب کنید اما این مسئله از این هم سریع‌تر می‌تواند حل شود :) اگر علاقه و وقت داشتید جا دارد تا چند ساعت روی این مسئله فکر کنید اگر هم هر یک را نداشتید در ادامه به حل می‌پردازیم.

ایده

خب با این ایده هم کمی به مسئله فکر کنید.

برویم سراغ حل. زیردنباله‌ای که جواب مسئله است، انتهایش یا روی اندیس ۰ است یا روی اندیس ۱ یا ... و یا روی اندیس n−1n−1. برای حالت‌بندی روی این انتهای زیردنباله، رویش فور می‌زنیم و این اندیس را jj در نظر می‌گیریم. حال فرض کنید مینیمم p[i]p[i]ها به ازای i<j≤0i<j≤0 را داریم. در این صورت بیشینه زیردنباله‌ای که به اندیس jj ختم می‌شود برابر است با p[i]−min(p[i])p[j]−min(p[i]). اگر بخواهیم مقدار مینیمم p[i]p[i]ها را هر سری محاسبه کنیم حلقه تودرتو داریم و زمان اجرا از O(n2)O(n2) خواهد بود. اما دوای این درد را قبلا هنگام تبدیل راه‌حل اول به دوم دیده‌ایم. اینجا هم درست به همان صورت وقتی jj زیاد می‌شود نیاز نیست که ما مینیمم p[i]p[i]ها را هر بار از اول حساب کنیم. بلکه کافی است مقدار آن را با تک عنصر جدید که به محدوده اضافه شده است (یعنی p[j−1]p[j−1]) مقدار مینیمم p[i]p[i]ها را به‌روزرسانی کنیم. به این ترتیب به الگوریتمی از چه اُردری می‌رسیم؟

اُردر

در صورت علاقه پس از گرفتن نمره کامل زیرمسئله، سعی کنید با تنها یک آرایه راه‌حل را پیاده‌سازی کنید.