مسأله‌ی تناسب: تبدیل رشته‌ی A به B با درج، حذف یا تعویض کاراکترها؛ کم‌ترین هزینه برای این کار را بیابید.

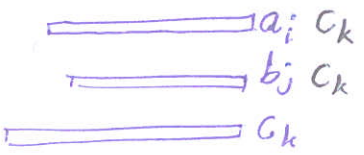
در این مسأله نیز مانند مسأله‌ی تناسب باید حالات مختلف را برای این ۳ رشته در نظر بگیریم. سه رشته‌ی A, B, C در اینجا مطرح شده‌اند:

Dynamic Programming:

تعریف می‌کنیم:

$$A(i) = a_1, a_2, \dots, a_i$$
$$B(j) = b_1, b_2, \dots, b_j$$
$$C(k) = c_1, c_2 \dots c_k$$

حال مانند مسأله‌ی تناسب حالات بازگشتی می‌نویسیم. $C(i,j,k)$ را به صورت بازگشتی می‌نویسیم. حالات زیر از حالاتی است که امکان دارد رخ دهد:



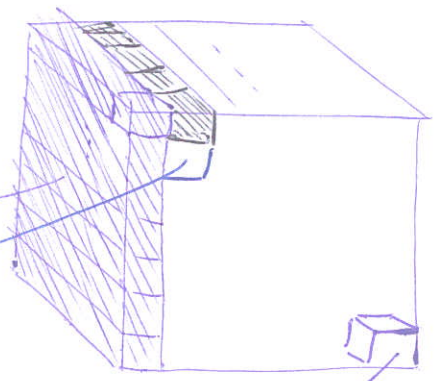
اگر کاراکتر c_k را به هر یک از دو رشته‌ی دیگر اضافه نکنیم، می‌توانیم بنویسیم:

$$C(i,j,k) = C(i,j,k-1) + 1$$

که ۱ به دلیل هزینه‌ی امتیازدهی مطرح شده در صورت سؤال است.

با در نظر گرفتن حالات مختلف و نوشتن کامل $C(i,j,k)$ می‌توانیم جدول DP را پر کرده که در این مسأله سه بعدی است.

مقداردهی اولیه نیز با دانستن A, B, C امکان‌پذیر است.



بررسی مسأله‌ی دوم :

روش اول : روش آسان حل این سؤال از مرتبه‌ی $O(n^2)$ بوده و با برنامه‌نویسی پویا انجام می‌شود.

یک جدول در نظر می‌گیریم که درآیه‌ی (i,j) آن معرف زیرمجموعه‌ای از مجموعه‌ی $\{x_1, x_2, \dots, x_j\}$ است که (یا زیرمجموعه‌هایی)

مجموع اعضای آن به پیمانه‌ی n برابر i است. حال دو حالت محتمل است رخ بدهد :

1) x_j در زیرمجموعه‌ی مذکور وجود نداشته باشد که در این صورت $C(i, j-1)$ را باید پیدا کرد.

2) x_j در زیرمجموعه‌ی گفته شده وجود دارد که داریم :

$$\sum_{k=1}^{j} x_k \overset{n}{\equiv} i \implies \sum_{k=1}^{j-1} x_k \overset{n}{\equiv} i - x_j$$

یعنی باید $C(i-x_j, j-1)$ را به دست آورد.

پس می‌توان جدول را به صورت بازگشتی پر کرده و $C(0,n)$ را که جواب مورد نظر ماست به دست آورد.

روش دوم : این روش از مرتبه‌ی $O(n)$ است.

ابتدا $Sum(i)$ را به صورت زیر تعریف می‌کنیم :

$$Sum(i) = x_1 + x_2 + \dots + x_i$$

برای همه‌ی $Sum(i)$ ها باقیمانده بر n را پیدا می‌کنیم. این کار از مرتبه‌ی $O(n)$ طول می‌کشد چون وقتی $Sum(i)$ را به دست می‌آوریم داریم :

$$Sum(i+1) = Sum(i) + x_{i+1}$$

حال در یک آرایه n عضوی به ازای هر باقیمانده یک خانه اختصاص می‌دهیم. به ازای هر خانه‌ی این آرایه، خانه‌ای در آرایه‌ای دیگر نیز برای ذخیره‌ی اندیس $Sum(i)$ مربوطه در نظر می‌گیریم. در حین به دست آوردن باقیمانده‌های $Sum(i)$

این دو آرایه را پر می‌کنیم. دو حالت مورد نظر است: 1) اگر خانه‌ی اول آرایه‌ی باقیمانده‌ها (که مربوط به باقیمانده‌ی صفر است) پر شد، اندیس مربوط بیانگر جواب مسأله است. 2) اگر یک خانه از آرایه برای بار دوم پر شد، در این صورت دو زیرمجموعه داریم که دارای باقیماندگی یکسانی هستند. یکی به صورت $\{x_1, x_2, \dots x_k\}$ و دیگری $\{x_1, x_2, \dots, x_l\}$، اعضای غیرمشترک این دو جواب مسأله‌اند.

بررسی مسأله‌ی سوم:

در حالت کلی مسأله را بررسی می‌کنیم:

$$S_1 = \{a_1, a_2, \dots, a_m\}$$

$$S_2 = \{b_1, b_2, \dots, b_n\}$$

ابتدا یکی از مجموعه‌ها مرتب می‌کنیم (این کار را برای S_2 انجام می‌دهیم که از مرتبه‌ی $O(n\log n)$ خواهد شد)

سپس به ازای هر عضو در S_1، مثلاً a_i، اگر عضوی در S_2 موجود باشد که مجموعشان با a_i، x شود داریم:

$$a_i + k = x \rightarrow k = x - a_i$$

پس می‌توانیم با binary search در S_2 این مقدار را پیدا کرده و عضوی که S_2 در غیر این صورت به سراغ a_2 رفت. این کار در کل از مرتبه‌ی $O(m\log n)$ است.

پس در مجموع این کار از مرتبه‌ی $O(n\log n)$ انجام می‌شود.

بررسی مسأله‌ی چهارم:

$$x_1, x_2, \dots, x_n \qquad \sum w_i = W$$
$$w_1, w_2, \dots, w_n$$

$$W \geq X > 0$$

پیدا کردن x_i به نحوه‌ای که:

$$\textcircled{1} \sum_{x_i > x_j} w_i < X \quad , \quad \textcircled{2} \sum_{x_i > x_j} w_i + w_j \geq X$$

ابتدا این دنباله‌ی اعداد را در $O(n\log n)$ مرتب می‌کنیم. برای راحتی کارهایمان نام‌گذاری عناصر را روی اعضا انجام می‌دهیم.

$$x_1 < x_2 < x_3 < \cdots < x_n$$

x_1 را چک می‌کنیم:

① $w_2 + w_3 + \cdots + w_n < X$

② $w_1 + w_2 + w_3 + \cdots + w_n \geq X$

این شرط به دلیل این‌که $W \geq X$ برقرار است.

در صورتی که ① برقرار نباشد داریم:

$$w_2 + w_3 + \cdots + w_n \geq X$$

پس برای چک کردن x_2 می‌دانیم که شرط ② را دارا است. پس رابطه‌ی X، $w_3 + \cdots w_n$ را برآورده می‌کند.

این چک کردن‌ها از مرتبه‌ی $O(n)$ قابل انجام است پس این روش از مرتبه‌ی $O(n\log n)$ انجام می‌گیرد.

روش دومی هم می‌توان برای این مسأله در نظر گرفت که از مرتبه‌ی $O(n)$ است.

برای این دنباله می‌توانیم میانه را در $O(n)$ پیدا کرد. اگر X_M را میانه در نظر گیریم و روی میانه شرط ② را چک کنیم

اگر $X \geq \sum_{x_j > X_M} w_j$ برقرار بود، در نیمه‌ی بزرگ‌تر باید بگردیم و در غیر این صورت در نیمه‌ی اول می‌گردیم.

اگر مرتبه‌ها را در نظر بگیریم داریم:

$$O(n) + O\left(\frac{n}{2}\right) + \cdots = \sum_{i=1}^{\log n} O\left(\frac{n}{2^i}\right) = \boxed{O(n)}$$

بررسی مسأله‌ی پنجم:

ابتدا رأس هم‌ارز را بررسی می‌کنیم اگر کوچک‌تر یا مساوی x بود به عمق عمل الگوریتم true داده و روی نودهای همسایه DFS را اجرا می‌کنیم و یک شمارنده تعریف می‌کنیم. اگر counter مقداری کوچک‌تر از k شد و الگوریتم هم به برای اعضای بزرگ‌تر از x

پایان رسید پس کمتر از k عدد داریم که از x بزرگترند یعنی kامین عضو بزرگ قطعاً کوچکتر یا مساوی x است.

اگر $Counter \geq k$ بود، در این حالت kامین عضو بزرگ حتماً از x بزرگتر است.

این الگوریتم از مرتبه‌ی $O(k)$ است.

کلاس حل تمرین آنالیز الگوریتم

الگوریتم kruskal

1) کوچکترین یال را در نظر بگیرید (از نظر وزن) و آن را e_1 بنامید.

2) فرض کنید $e_1, \dots, e_i$ مشخص باشد؛ e_{i+1} یالی است که با $G(e_1, \dots, e_i)$ تشکیل دور نمی دهد، و $w(e_{i+1})$ نیز مینیمم مقدار را دارد (از بین یال های انتخاب نشده)

3) گام 2) را آنقدر ادامه دهید تا الگوریتم به اتمام برسد.

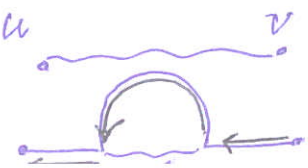
اثبات اینکه این الگوریتم درخت بهینه را تولید می کند:

T را درخت حاصله از الگوریتم kruskal، T' را درخت ایده آل و بهینه فرض می کنیم. حال فرض کنید در حین ساختن T تا مرحله $i-1$ ام همگی یال ها بین T و T' مشترک باشند و e_i را اولین یالی می دانیم که در T' وجود ندارد. چون T' درخت است، افزودن e_i ایجاد دور می کند پس شکل زیر را داریم:

باید یالی باشد که در T نباشد ولی در T' باشد چون در غیر این صورت در T' دور خواهیم داشت پس اگر این یال را e_i بنامیم و آن را از T' حذف کنیم، گراف جدیدی داریم که آن را T'' می نامیم

$$T'' = (T' + e_i) \setminus e_i$$

هر دو رأس مانند u, v یا دارای مسیری روی T هستند که e_i در آن مسیر وجود ندارد یا اینکه با اتصال e_i می توان مسیر جدیدی بین u و v ایجاد کرد پس T'' همبند است و چون $i-1$ یال دارد پس درخت است.



وزن T'' را به صورت زیر می‌توان نوشت :

$$w(T'') = w(T') + w(e_i) - w(e_i')$$

اما در انجام روش کروسکال e_i را یالی برگزیده‌ایم که وزن آن از همه‌ی یال‌هایی که به درخت در مرحله‌ی i ام وصل نشده‌اند کوچکتر است پس این کوچکتر بودن شامل e_i' هم می‌شود یعنی :

$$w(e_i) \leqslant w(e_i')$$

که نتیجه می‌دهد :

$$w(T'') \leqslant w(T')$$

که به دلیل بهینه بودن T' داریم :

$$w(T'') = w(T')$$

پس T'' هم وزن را دارد . با این استدلال می‌توان T'' را با استفاده از کروسکال بسترش و درختی بهینه با این روش تولید کرد .

$O(kruskal) = ?$

$O(E \log E)$ ⟵ مرتب کردن یال‌ها براساس وزن

چون در تولید درخت این روش ، از یک جنگل ، به درخت می‌رسیم پس نیاز داریم مؤلفه‌های همبندی را اندکی کنیم یا اینکه دو رأس یک یال در یک مؤلفه‌ی همبندی قرار دارد یا نه را چک کنیم پس می‌توان با استفاده از آشاره‌گر برای هر رأس این کار را انجام داد . برای رأسی که عضو هیچ مؤلفه‌ای نباشد هم Null را در نظر می‌گیریم (پس چک کردن در $O(1)$ انجام می‌شود)

$$E \leqslant V^2 \Longrightarrow \log E \leqslant 2 \log V \Longrightarrow \boxed{O(E \log E) = O(E \log V)}$$

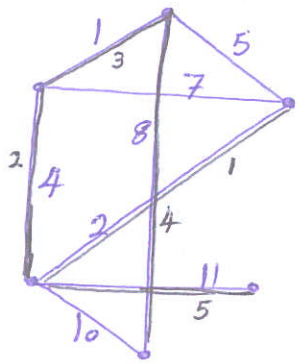
الگوریتم Prim :

1) یک رأس دلخواه انتخاب کنید و یکی از یال‌های وابسته به آن که مینیمم وزن را دارد، انتخاب کنید.

2) فرض کنید الگوریتم تا گام i ام، زیردرخت $G[e_1, \ldots, e_i]$ باشد. یال e_{i+1} را از بین تمام یال‌هایی که یک سر آن‌ها متصل به زیرگراف G است و سر دیگر به بقیه‌ی رأس‌ها متصل است، انتخاب می‌کنیم به طوری که $w(e_{i+1})$ مینیمم شود.

3) الگوریتم را ادامه می‌دهیم تا $n-1$ یال پدید آمده باشد.

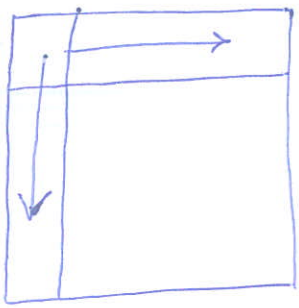
مثال:

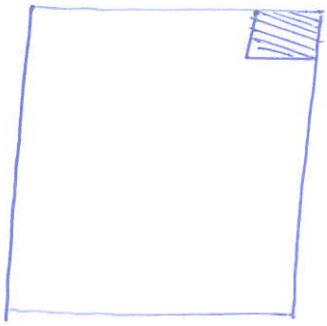


می‌خواهیم Prim را بر روی این گراف پیاده کنیم. اعداد با رنگ سیاه ترتیب انتخاب یال‌ها را نشان می‌دهند.

مسأله‌ی امتحانی:

فرض کنید A یک ماتریس $m \times n$ است که اعداد هر سطر و ستون آن صعودی اند. الگوریتمی پیدا کنید که بگوید عدد x در این ماتریس وجود دارد یا نه:





از آخرین خانه‌ی سطر اول شروع می‌کنیم. اگر این عدد از x بزرگتر باشد که در این صورت کل ستون زیر آن نیز از x بزرگتر است و آن ستون را حذف می‌کنیم و به عدد قبل آن در همین سطر می‌رویم. و اگر نه در صورتی که از x کوچکتر باشد یعنی سطری که در آن قرار دارد، قطعاً از x کوچکتر نیست در این ستون به پایین می‌رویم (یا هر سؤال یک سطر یا ستون حذف می‌شود). حال اگر ما وی x بودیم که به جواب رسیده‌ایم وگرنه $O(m+n)$ می‌زنیم به جواب رسید.