مسیر دو طرفه کامل(مسئله ساده)

فرض کنید یک گراف همبند و غیر جهت دار $G(V,E)$ داریم. میخواهیم یک مسیر پیدا کنیم که **تمام یال های گراف را دقیقا یکبار در هر دو جهت پیمایش کند**. مثلا اگر بین راس های u و v یک یال وجود داشته باشد، مسیر پاسخ باید شامل دقیقا یک u→v و دقیقا یک v→u باشد. برای پیدا کردن این مسیر یک الگوریتم با پیچیدگی زمانی $O(V+E)$ پیشنهاد کنید.

حل:

گام اول: پیدا کردن یک درخت و ترتیب گذاری.

ابتدا با استفاده از الگوریتم BFS،یک نود را به صورت رندوم به عنوان ریشه انتخاب کرده و یک درخت داخل گراف G تشکیل میدهیم. همچنین هنگام اجرای الگوریتم به هر راس یک عدد اختصاص می دهیم که نمایانگر فاصله آن از ریشه است(برای این کار کافی است هنگام رسید به هر نود جدید، فاصله پدر آن نود + ۱ را در آن قرار دهیم). این ترتیب گذاری بعدا به ما کمک می کند تا از پیموده شدن دوباره مسیر جلوگیری شود. درخت ایجاد شده را T می نامیم.

گام دوم: ایجاد مسیر

در ادامه از شبه کد زیر استفاده میشود :

```
make_path(u,G,T)
   for v adjacent to u in the G, but not in the T such that u.num ≤ v.num:
       go to v and back to u
   for v adjacent to u in the T except the parent of u:
       go to v
       make_path(u,G,T)
   if u is not the root of T:
       return # back to the parent of u
   else
       finish
```

و نهایتا به ورودی u تابع، T.root را می دهیم.

گام سوم: تحلیل زمانی

1. مرحله اول: اجرای الگوریتم BFS $O(V+E)$

2. مرحله دوم:

- حلقه اول در مجموع تمام اجرا شدن های بازگشتی $O(E)$ بار اجرا می شود زیرا روی هر یال دقیقا یک بار رفته می شود و برگشته می شود.
- حلقه دوم نیز در مجموع $O(V)$ بار اجرا می شود. زیرا بر روی هر راس دقیقا یکبار اجرا میشود(حرکت بر روی درخت به شکل depth-first)
- بازگشت به نود پدر نیز $O(V)$ بار اجرا میشود.

بنابراین نهایتا خواهیم داشت:

$$O(V + E) + O(E) + O(V) = O(V + E)$$

مسئله گل کاری (مسئله پیچیده)

یک کشاورز میخواهد تا در زمین خود گل بکارد. برای این کار او n بذر گل که به صورت $(1, 2, 3, \ldots, n)$ نامگذاری شده اند دارد. همچنین زمین او فقط جا برای k گل دارد. آقای کشاورز نمیخواهد حتی یک جای خالی در زمینش وجود داشته باشد، بنابراین او **دقیقا باید k گل بکارد یا اصلا گل نکارد!** همچنین برخی از گل ها نمیتوانند از همدیگر جدا باشند. خوشبختانه آقای کشاورز میداند کدام گل ها باید باهم دیگر کاشته شوند. آنها را با

E = $\{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$

نشان میدهیم که یک مجموعه از جفت گل هایی است که باید با همدیگر کاشته شوند. هر جفت (i,j) نشان میدهد که گل i و گل j باید با همدیگر کاشته شوند(نمی توانند جدا باشند). **بنابراین کشاورز یا باید هر دوی آنها را بکارد و یا هیچ کدام را نکارد**. آیا میتوانید به کشاورز کمک کنید و به او بگویید که آیا میتواند گل بکارد یا نه ؟ اگر می تواند گل هایی که باید بکارد را نشان دهید.

یک الگوریتم طراحی کنید که با استفاده از 3 ورودی k , n و E بتواند در پیچیدگی زمانی O(m+nk) بگوید که آیا کشاورز می تواند گل بکارد یا نه (اگر می تواند آن گل ها را پیدا کنید). سپس درستی الگوریتم خود را بیان اثبات کنید و پیچیدگی زمان آن را نیز بدست آورید. (واضح است که اگر n < k باشد باشد، این امکان وجود ندارد.)

پاسخ:

گام اول: فرموله کردن مسئله

مسئله را به یک گراف G تبدیل می کنیم که در آن راس ها گل ها هستند و هر جفت گل در مجموعه E که باید با هم کاشته را به یک یال در گراف G تبدیل میکنیم.

گام دوم: پیدا کردن مولفه های همبند

حالا یکی از الگوریتم های BFS یا DFS را بر روی گراف G اجرا می کنیم تا مولفه های همبند را پیدا کنیم. پس از آن ما x مولفه همبند با سایز های $c_1, c_2, \ldots, c_x$ خواهیم داشت. به طوری که

$$\sum_{i \in [1,x]} c_i = n$$

بدیهی است که تمام گل هایی که در یک مولفه همبند قرار دارند باید با یکدیگر برداشته شوند یا هیچکدام از آنها برداشته نشوند. (در غیر اینصورت طبق تعریف گفته شده از یال، حداقل یک جفت گل وجود دارند که کنار هم نیستند.)

گام سوم: استفاده از dynamic programing برای حل مسئله

آرایه D[i,j] را طوری تعریف میکنیم که D[i,j] = 1 اگر یک زیر مجموعه از $c_1, c_2, \dots, c_i$ وجود داشته باشد که جمع آنها برابر با j باشد. در غیر اینصورت (هیچ زیر مجموعه ای وجود نداشته باشد که جمع آنها برابر با j شود) آنگاه D[i, j] = 0 است.

همچنین :

D[i, j] = 0 if j < 0 or j > k (بیرون از جدول)

حالات پایه:

D[i, 0] = 1 for all i ∈ [0, x] (چون در این حالت می توانیم هیچ گلی را انتخاب نکنیم)
D[0, j] = 0 for all j ∈ [1, k]

نحوه محاسبه هر خانه:

D[i, j] = max(D[i-1, j] ,D[i-1, j - c_i])

که در واقع انتخاب می کنیم که مولفه c_i انتخاب شود یا نشود.

جواب نهایی:

جواب نهایی مسئله در D[x,k] خواهد بود.

گام چهارم: اثبات درستی

اثبات را با استقرا روی i انجام می‌دهیم.

حالت پایه: وقتی i=0 باشد، مقداردهی اولیه ماتریس به وضوح صحیح است، زیرا D[0,0] = 1 و برای هر j>0 مقدار D[0,j]=0 است. این بدین معناست که وقتی هیچ مؤلفه‌ای در نظر گرفته نشده است، تنها مجموعی که می‌توان به آن رسید مقدار صفر است.

گام استقرا: فرض می‌کنیم که برای همه‌ی مقادیر j، مقدار D[i-1, j] به درستی محاسبه شده است. حال می‌خواهیم نشان دهیم که مقدار D[i, j] نیز به درستی محاسبه می‌شود.

- اگر D[i, j] = 0، پس طبق رابطه‌ی بازگشتی داریم:
  $D[i-1, j] = 0 \; and \; D[i-1, j-c_i] = 0$
  یعنی هیچ زیرمجموعه‌ای از مؤلفه‌های $\{c_1, c_2, \dots, c_{i-1}\}$ وجود ندارد که مجموع اندازه‌های آن برابر j یا $j - c_i$ باشد. بنابراین، اگر زیرمجموعه‌ای از $\{c_1, c_2, \dots, c_{i-1}\}$ وجود داشت که مجموع اندازه‌های آن برابر j باشد، دو حالت ممکن بود:

1. اگر c_i در این زیرمجموعه باشد، در این صورت مجموعه‌ی باقی مانده باید مجموعی برابر $j - c_i$ داشته باشد، که طبق فرض استقرا امکان‌پذیر نیست.
2. اگر c_i در این زیرمجموعه نباشد، پس همان زیرمجموعه باید در مجموعه‌ی $\{c_1, c_2, \dots, c_{i-1}\}$ موجود باشد، اما طبق فرض استقرا این نیز ممکن نیست.

در نتیجه، مقدار D[i, j] = 0 به درستی محاسبه شده است.

- اگر D[i, j] = 1، پس حداقل یکی از دو مقدار $D[i-1, j]$ یا $D[i-1, j-c_i]$ برابر 1 است:

1. اگر $D[i-1, j] = 1$، پس زیرمجموعه‌ای از $\{c_1, c_2, \dots, c_{i-1}\}$ با مجموع j وجود دارد. از آنجا که این زیرمجموعه در $\{c_1, c_2, \dots, c_i\}$ نیز وجود دارد، مقدار D[i, j] = 1 صحیح است.
2. اگر $D[i-1, j-c_i] = 1$، پس زیرمجموعه‌ای از $\{c_1, c_2, \dots, c_{i-1}\}$ وجود دارد که مجموع آن برابر $j - c_i$ است. در این حالت، می‌توان c_i را به این زیرمجموعه اضافه کرد تا مجموعه‌ای با مجموع j به دست آید. این نیز نشان می‌دهد که مقدار D[i, j] به درستی برابر 1 قرار داده شده است.

نتیجه: از آنجا که هدف ما یافتن یک زیرمجموعه از مؤلفه‌های متصل است که مجموع اندازه‌های آن برابر k باشد، مقدار D[x,k] تعیین می‌کند که آیا چنین انتخابی ممکن است یا خیر. اگر D[x,k] = 0 باشد، نمی تواند گل بکارد، در غیر این صورت، با دنبال کردن مقادیر محاسبه‌شده در ماتریس D، می‌توان مجموعه‌ی موردنظر را بازسازی کرد.

گام پنجم: محاسبه پیچیدگی زمانی

1. استفاده از BFS و یا DFS برای پیدا کردن مولفه های همبند: $O(m+n)$
2. حل DP:
   - تعداد زیر مسئله ها: $O(xk) = O(nk)$
     چون $x = O(n)$ بنابراین در جدول D $O(nk)$ خانه داریم.
   - حل هر زیر مسئله: $O(1)$
   - بدست آوردن جواب نهایی:
     1. بررسی امکان پذیر بودن: $O(1)$
     2. بازسازی جواب در صورت امکان پذیر بودن: $O(n)$
     از آنجا که هر مرحله از این بازسازی تنها یک گام به عقب در ماتریس حرکت می‌کند و تعداد این گام‌ها حداکثر x است، این فرآیند در زمان $O(n)$ انجام می‌شود.

بنابراین پیچیدگی نهایی برابر است با:

$$O(m+n) + O(nk) = O(m+nk)$$