مجموعه‌ای از مسائل طراحی الگوریتم‌ها
مبتنی بر اسلایدهای دکتر قدسی

عبّاس محرابیان

۲۰ تیر ۱۳۸۶

در این مسائل منظور از گراف در حالت کلّی گراف بدون جهت است.

۱ استقرا

مسئله‌ی ۱.۱. [۲] بزرگترین زیرگراف از یک گراف را به دست آورید که درجه‌ی هر راس در آن لااقل k باشد.

ایده‌ی حل: اگر درجه‌ی همه‌ی رئوس لااقل k باشد که مسئله حل است در غیر این صورت یک رأس با درجه کمتر از k را یافته آن‌را از گراف حذف کنید و استقرا بزنید.

مسئله‌ی ۲.۱. [۲] n عدد حقیقی $x_1, \ldots, x_n$ داده شده‌است. اعداد $1 \leq i \leq j \leq n$ را پیدا کنید که حاصل $x_i + x_{i+1} + \ldots + x_j$ بیشینه شود.

ایده‌ی حل: اعداد را یکی یکی بررسی کنید و در هر لحظه بیشترین جمع موجود و بیشترین جمع موجود که مختوم به آخرین عدد باشد را نگه دارید. شبه کد زیر حاصل $x_i + x_{i+1} + \ldots + x_j$ را می‌دهد:

```
sm = gm = 0
for i = 0 to n do
    if x[i] + sm > gm then
        sm = sm + x[i]
        gm = sm
    else
        if x[i] + sm > 0 then
            sm = sm + x[i]
        else
            sm = 0
return gm
```

مسئله‌ی ۳.۱. [۲] n عدد داریم و می‌خواهیم تعدادی از آنان را بیابیم که جمعشان k بشود. یک جعبه سیاه داریم که با گرفتن تعدادی عدد و عدد x می‌گوید آیا جمع تعدادی از آن اعداد x می‌شود یا خیر. با فرض وجود جواب الگوریتمی ارائه دهید که با n بار استفاده از این جعبه سیاه، زیرمجموعه‌ی مطلوب را بیابیم.

ایده‌ی حل: برای هر عدد می‌توان با یک بار استفاده فهمید که در آن زیرمجموعه هست یا نه.

۲ تقسیم و حل

مسئله‌ی ۱.۲. [۲] اعداد متمایز $a_1 < a_2 < \ldots < a_n$ داده شده‌اند. الگوریتمی ارائه دهید تا مشخص کند آیا $1 \leq i \leq n$ ای با خاصیت $a_i = i$ داریم؟

ایده‌ی حل: از binary search استفاده کنید.

مسئله‌ی ۲.۲. [۲] قطر یک درخت دودویی را در زمان خطی بیابید.

ایده‌ی حل: DFS بزنید، برای هر راس ابتدا ارتفاع زیردرخت چپ و راستش را بیابید سپس اگر مجموع این دو از قطر فعلی بهتر است قطر را به‌روز کنید سپس ارتفاع خود این راس را مقداردهی کنید و برگردید.

مسئله‌ی ۳.۲. [۲] در $O(n^۲)$ یک آرایه را پُر کنید که خانه‌ی (i,j) آن نشان دهنده‌ی فاصله‌ی رئوس v_i و v_j از یک درخت دودویی n رأسی داده شده است.

ایده‌ی حل: همانند مسئله قبل DFS بزنید و قبل از بازگشت تمام فواصل بین رئوس زیردرخت‌های چپ و راست رأس فعلی را مقدار دهید.

۳ تحلیل سرشکن شده

مسئله‌ی ۱.۳. [۱] یک پشته را با دو صف پیاده‌سازی کنید که هزینه‌ی سرشکن اعمال درج و حذف $O(۱)$ باشد.

ایده‌ی حل: دو صف را A و B بنامید. آن‌گاه اعمال درج (Enqueue) و حذف (Dequeue) به این طریق میسر است:

```
Enqueue(x)
    push(A, x)

Dequeue()
    if B is empty then
        while A is not empty do
            push(B, pop(A))
    return pop(B)
```

مسئله‌ی ۲.۳. [۱] نشان دهید اگر به شمارنده‌ی دودویی عمل Reset را هم اضافه کنیم باز هم هزینه‌ی سرشکن همه‌ی اعمال $O(۱)$ است.

ایده‌ی حل: در هر لحظه در متغیری نگه دارید پرارزش‌ترین بیت ۱ کجاست و برای Reset آن متغیر را صفر کنید، عمل Increment را نیز باید تغییر دهید.

مسئله‌ی ۳.۳. [۱] نشان دهید اگر عمل Decrement را نیز به شمارنده اضافه کنیم دیگر هزینه‌ی سرشکن لزوماً $O(1)$ نیست. ولی اگر از سیستم نمایشی استفاده کنیم که در آن هر رقم می‌تواند ۰ یا ۱ یا ۲ باشد می‌توان این دو عمل را طوری پیاده‌سازی کرد که هزینه‌ی سرشکن هر دو ثابت باشد. در این سیستم نمایش داریم مثلاً: $۱۸ = (۱۰۰۱۰) = ۰ \times ۲^۰ + ۱ \times ۲^۱ + ۲ \times ۲^۲ + ۱ \times ۲^۳ = (۱۲۱۰)$

ایده‌ی حل: برای قسمت اول شمارنده را برابر $۲^{n-۱} - ۱$ قرار دهید و متناوباً افزایش و کاهش دهید.

مسئله‌ی ۴.۳. [۱] داده‌ساختاری از اعداد را معرفی کنید که عمل درج یک عنصر و عمل حذف $n/۲$ عنصر بزرگ را به صورت سرشکن در زمان ثابت انجام دهد.

ایده‌ی حل: از یک آرایه معمولی استفاده می‌کنیم، برای حذف نصفه‌ی بزرگ، میانه را در $O(n)$ پیدا می‌کنیم. برای تحلیل، تابع پتانسیل را دو برابر تعداد اعضای موجود تعریف می‌کنیم.

۴ برنامه‌ریزی پویا

مسئله‌ی ۱.۴. [۲] n عدد داریم که مجموعشان S است. الگوریتمی از $O(nS)$ ارائه دهید که بگوید آیا می‌توان اعداد را به دو دسته با مجموع مساوی تقسیم کرد یا خیر.

ایده‌ی حل: همان مسئله کوله‌پشتی را برای $S/۲$ حل می‌کنیم.

---

[۱] Problem Set 3 of *CS 423: Theory of Algorithms*, Princeton University, Spring 2000

مسئله‌ی ۲.۴. [۲] اعداد متمایز $s_1, \ldots, s_n$ داده شده‌اند. طول بزرگ‌ترین زیردنباله‌ی اکیداً صعودی آن‌ها را در $O(n^2)$ بیابید.

ایده‌ی حل: l_i را برابر طول بزرگ‌ترین زیردنباله‌ی صعودی مختوم به s_i می‌گیریم. در این‌صورت $l_0 = 0$ و $l_i = max\{l_j : 0 \leq j < i \text{ and } s_j < s_i\} + 1$ و جواب مسئله $max\{l_i\}$ است.

مسئله‌ی ۳.۴. [۲] همان مسئله قبل را در زمان $O(n\ logn)$ حل کنید.

ایده‌ی حل: یک زیردنباله‌ی صعودی را بهتر از دیگری می‌نامیم اگر آخرین عنصرش از دیگری کوچک‌تر باشد. در هر لحظه بهترین زیردنباله‌های صعودی با طول‌های $1, 2, \ldots, l$ را نگه دارید، که l طول بزرگ‌ترین زیردنباله‌ی صعودی به دست آمده تا این لحظه است. سپس با بررسی هر عدد جدید، دقیقاً یکی از این زیردنباله‌های خوب تغییر می‌کند، یا یک زیردنباله‌ی جدید به طول $l + 1$ اضافه می‌شود. در شبه‌کد زیر، آرایه‌ی last اندیس آخرین عنصر بهترین زیردنباله‌های صعودی را نگه می‌دارد و از آرایه‌ی prev می‌توان برای بازسازی زیردنباله‌ها استفاده کرد.

```
Input: Array x[1..n] of numbers
Output: Array ans[1..l] the Longest strictly Increasing Subsequence of x

LsIS(x)
l = 1
last[1] = 1
for i=2 to n do
     if x[i] < x[last[1]] then
        last[1] = i
        prev[i] = 0
     else if x[i] > x[last[l]] then
        l = l + 1
        last[l] = i
        prev[i] = last[l-1]
     else
        find maximum k with x[last[k]] < x[i] < x[last[k+1]] using binary search
```

```
        last[k+1] = i
        prev[i] = last[k]

s = last[l]
for i=l to 1 do
    ans[i] = x[s]
    s = prev[s]
return ans
```

مسئله‌ی ۴.۴. همه‌ی رشته‌های دودویی به طول n که حداکثر L تا ۱ دارند را به ترتیب الفبایی مرتب کرده‌ایم. kمُین این رشته‌ها را در زمان چندجمله‌ای بیابید.

ایده‌ی حل: $d[i][j]$ را برابر تعداد رشته‌های دودویی به طول i که حداکثر j تا ۱ دارند تعریف کنید. مقادیر d را می‌توان به کمک رابطه‌ی زیر به صورت پویا پر کرد: $d[i][j] = d[i-1][j] + d[i-1][j-1]$. پس از پُر کردن این آرایه، با فراخوانی تابع find(k,n,L) جواب (به صورت برعکس) در آرایه ans قرار می‌گیرد. به شبه کد زیر توجه کنید:

```
Returns the k-th element of the set of 0-1 strings with length i and at most j 1's
find(k,i,j)
    if i=0 then return
    if d[i-1][j] < k then
        ans[i] = 1
        find(k-d[i-1][j], i-1, j-1)
    else
        ans[i] = 0
        find(k,i-1,j)
```

مسئله‌ی ۵.۴. [۲] یک رشته از حروف بزرگ داریم و می‌خواهیم آن‌را کدگذاری کنیم.

[۲] ACM-ICPC North-Eastern European Regional Contest 2002-03, http://neerc.ifmo.ru/past/2002.html

در کد مربوطه، عبارتی مانند m(S) به m تا رشته‌ی S دیکد می‌شود که در آن m عددی طبیعی (نوشته شده در مبنای ده) است. خود S ممکن است عبارتی کدشده باشد. به مثال‌های زیر توجه کنید: $۳(AB) \rightarrow ABABAB$

$۲(AB۴(C)۳(D۱(A))) \rightarrow ABCCCCDADADAABCCCCDADADA$

الگوریتمی از $O(n^{۳.۵})$ ارائه کنید که رشته‌ی S به طول n را به صورت بهینه کد کند.

ایده‌ی حل: $S[i,j]$ را برابر زیررشته‌ای از S تعریف کنید که از کارکتر iمُ شروع شده و تا jمُ ادامه می‌یابد. پس $S = S[۱,n]$. $d[i,j]$ را برابر طول کمترین کد لازم برای کد کردن $S[i,j]$ تعریف کنید. جواب برابر $d[۱,n]$ است. اگر بتوان این قسمت را به صورت $k(R)$ کد کرد، آن‌گاه باید داشته‌باشیم $S[i,j] = RR\ldots R$ و و این که آیا $S[i,j]$ نمایشی این چنینی دارد یا خیر را می‌توان در $O(j-i+۱)$ بررسی کرد. تعداد kهایی که باید امتحان کرد حداکثر $O(\sqrt{n})$ است زیرا $k|j-i+۱$. تعداد ارقام k را نیز باید محاسبه کرد. مینیمم کد لازم برای R قبلاً محاسبه شده است. پس چون $۱ \leq i \leq j \leq n$ هزینه‌ی این قسمت $O(n^{۳.۵})$ است.

این احتمال را نیز باید درنظر گرفت که $S[i,j]$ بدین صورت کد نشود. پس باید $S[i,j]$ را برابر مینیمم مقداری که از این روش به دست می‌آید و $min\{d[i,k]+d[k+۱,j] : i \leq k < j\}$ قرار داد. هزینه‌ی این کار در کل $O(n^{۳})$ است.

مسئله‌ی ۶.۴. [۳] یک متن داریم که در آن از نمادهایی از زبان‌های مختلف استفاده شده‌است، و می‌خواهیم این رشته را کد کنیم. ولی تنوع کارکترهایی که می‌توانیم در کد از آن‌ها استفاده کنیم کم است و نمی‌توانیم به هر نماد از هر زبان یک کارکتر متمایز نسبت دهیم. یعنی برای زبان‌های متفاوت مجبوریم از مجموعه کارکترهای یکسان استفاده کنیم، ولی می‌توانیم به دیکودر بگوئیم در حال حاضر از چه زبانی استفاده می‌کنیم و دیکودر با توجه به زبان، این کارکتر را به یک نماد مناسب از آن زبان دیکود می‌کند. کارکترهایی که در کد استفاده می‌شوند دو دسته‌اند: کارکترهای معمولی متناظر با یک کارکتر از زبان فعلی هستند، و کارکترهای تغییر زبان، که دیکودر با دیدن آن‌ها زبان را تغییر می‌دهد. این کارکترها خود دو نوع‌اند: کارکتر «تغییر زبان لحظه‌ای» که به دیکودر اعلام می‌کند فقط کارکتر بعدی از زبان دیگری است، بعد از این کارکتر حتماً باید یک کارکتر معمولی بیاید. کارکتر «تغییر زبان دائمی» که به دیکودر اعلام می‌کند از این به بعد زبان عوض می‌شود. همه‌ی زبان‌ها کارکتر تغییر زبان دائمی را دارند، ولی تغییر زبان لحظه‌ای فقط به برخی از زبان‌ها ممکن است. بنابراین اگر طول متن n و تعداد زبان‌ها m باشد، حداکثر m کارکتر تغییر زبان لحظه‌ای و دقیقاً m کارکتر تغییر زبان دائمی داریم. الگوریتمی از $O(m^{۲}n)$ ارائه دهید که یک متن را به صورتی کد کند که تعداد کارکترهای کد مینیمم

باشد.

ایده‌ی حل: تعریف می‌کنیم $c[i,k]$ برابر است با مینیمم کارکترهای لازم برای کد کردن i کارکتر اول متن، به طوری که در آخر در زبان kمُ باشیم. برای پیدا کردن هر مقدار حداکثر باید m۲ حالت را درنظر بگیریم.

مسئله‌ی ۷.۴. [۳] الگوریتمی از مرتبه‌ی زمانی $O(n^{۲}۲^{n})$ برای مسئله‌ی فروشنده‌ی دوره‌گرد ارائه دهید.

ایده‌ی حل: اگر $C(i,j)$ هزینه‌ی رفتن از شهر i به شهر j باشد. $T(i;j_{۱},j_{۲},\ldots,j_{k})$ را برابر مینیمم هزینه‌ی سفری از i به ۱ تعریف کنید که طی آن به تمام شهرهای $j_{۱},j_{۲},\ldots,j_{k}$ دقیقاً یک بار رفته‌ایم. بنابراین جواب مسئله برابر $T(۱;۲,۳,\ldots,n)$ است و به طریق زیر قابل محاسبه است:

$$T(i;j_{۱},j_{۲},\ldots,j_{k}) = min_{۱\leq m\leq k}\ [C(i,j_{m}) + T(j_{m};j_{۱},\ldots,j_{m-۱},j_{m+۱},\ldots,j_{k})]$$

و شرط اولیه‌ی زیر را داریم: $T(i;j) = C(i,j) + C(j,۱)$.

مسئله‌ی ۸.۴. یک رشته‌ی S به طول l و n رشته با مجموع m داریم. می‌خواهیم طول بزرگ‌ترین پیشوندی از S را بیابیم که با این رشته‌ها قابل ساختن است. الگوریتمی از مرتبه‌ی $O(ml)$ برای این کار ارائه کنید.

ایده‌ی حل: $d(i)$ را برابر TRUE قرار دهید اگر و تنها اگر بتوان با آن رشته‌ها، پیشوند iمُ S را ساخت. مقداردهی $d(i)$ها به راحتی و شبیه مسئله‌ی کوله‌پشتی صورت می‌گیرد.

۵ الگوریتم‌های حریصانه

مسئله‌ی ۱.۵. [۳] فرض کنید m شیء با حجم معلوم داریم و می‌خواهیم تعدادی از آن‌ها را در جعبه‌ای به حجم معلوم قرار دهیم به طوری که بیشترین حجم ممکن از آن پر

[۳] Problem Set 7 of *Introduction to Algorithms*, MIT, November 2005

شود. می‌دانیم حجم جعبه از حجم هیچ شیئی کم‌تر نیست. اگر جواب بهینه‌ی مسئله برابر OPT باشد، نشان دهید هر یک از الگوریتم‌های زیر، لااقل به اندازه‌ی $OPT/۲$ جعبه را پر می‌کنند. سپس مرتبه‌ی زمانی آن‌ها را مقایسه کنید. در این الگوریتم‌ها منظور از «بررسی» یک شیء این است که می‌بینیم با توجه به اشیائی که تا الان در جعبه هستند، آن شیء در جعبه جا می‌شود و اگر جا می‌شد قرارش می‌دهیم.

- اشیا را از بزرگ به کوچک مرتب کرده سپس به همین ترتیب بررسی می‌کنیم.
- اشیا را یکی‌یکی نگاه می‌کنیم و آن‌هایی که حجمشان در بازه‌ی $[\frac{n}{۴}, n]$ است را بررسی می‌کنیم. در مرحله‌ی بعدی اشیائی که حجمشان در $[\frac{n}{۴}, \frac{n}{۲}]$ است بررسی می‌کنیم. این مراحل را انقدر تکرار می‌کنیم تا همه‌ی اشیا بررسی شده‌باشند.
- اشیا را یکی‌یکی نگاه می‌کنیم و آن‌هایی که حجمشان در بازه‌ی $[\frac{n}{۲}, n]$ است را بررسی می‌کنیم. در مرحله‌ی بعدی همه‌ی اشیا را بررسی می‌کنیم.

ایده‌ی حل: دو حالت بگیرید: شیئی با حجم بیشتر از نصف حجم جعبه وجود نداشته‌باشد، یا وجود داشته‌باشد. الگوریتم‌ها به ترتیب از مرتبه‌ی $O(mlog\ m)$،$O(mlog\ n)$ و $O(m)$ هستند.

مسئله‌ی ۲.۵. [۴] درختی داریم و می‌خواهیم روی تعدادی از رئوس و یال‌های آن دوربین قرار دهیم که تمام رئوس و یال‌های آن دیده‌شوند و از کم‌ترین تعداد دوربین استفاده کرده باشیم. دوربینی که روی رأس قرار می‌گیرد، یال‌ها و رئوس مجاور آن رأس را می‌بیند. دوربینی که روی یال قرار می‌گیرد، دو رأس مجاور و یال‌های مجاور این دو رأس را می‌بیند. الگوریتمی با زمان خطّی برای این کار ارائه دهید.

ایده‌ی حل: DFS بزنید و وقتی روی رأسی هستید قبل از بازگشت، طوری دوربین‌ها را قرار دهید که این رأس و یال‌هایی که آن‌را به فرزندان متصل می‌کنند دیده شوند، این کار حداکثر به پنج طریق ممکن است و باید با توجه به اطّلاعاتی که از فرزندان به دست آمده بهترین روشش به صورت حریصانه انتخاب شود. سپس این که از کدام روش استفاده شده را به عنوان اطّلاعات به رأس بالاتر انتقال دهید.

---

[۴] ACM-ICPC Sharif Preliminary Contest, 2006-07

٦ الگوریتم‌های پایه‌ی گراف

مسئله‌ی ۱.٦. [۲] گراف جهت‌دار G داده شده‌است. الگوریتمی با مرتبه‌ی زمانی $O(VE)$ ارائه دهید که تشخیص دهد آیا این گراف دارای دور جهت‌دار به طول فرد است یا خیر.

ایده‌ی حل: ابتدا G را به مولفه‌های قویاً همبند تقسیم کنید. از همه‌ی رئوس DFS بزنید و دنبال رأسی بگردید که از ریشه یک مسیر زوج و یک مسیر فرد به آن وجود داشته باشد، سپس از ویژگی قویاً همبندی استفاده کنید.

مسئله‌ی ۲.٦. رئوس گراف جهت‌دار G از ۱ تا n شماره‌گذاری شده‌اند. برای دو رأس i و j می‌نویسیم $i \longrightarrow j$ اگر مسیری جهت‌دار از i به j وجود داشته‌باشد. برای رأس i تعریف می‌کنیم $g(i) = max\{j : i \longrightarrow j\}$. الگوریتمی از $O(V+E)$ ارائه دهید که مقادیر $g(i)$ را برای $1 \leq i \leq n$ حساب کند.

ایده‌ی حل: گراف G^* را با برعکس کردن یال‌های G می‌سازیم. حال تعریف می‌کنیم $ind[i] = max\{j : j \longrightarrow i\}$ به وضوح مقدار $ind[i]$ در G^* با مقدار $g(i)$ در G برابر است. $ind[i]$ها را به کمک تابع زیر می‌یابیم:

```
global variables: father, ind[n] with initial value 0

findInds()
    for i=1 to n do
        father = i
        dfs(i)

dfs(v)
    for every child u of vertex v do
        if ind[u] = 0 then
            ind[u] = father
            dfs(u)
```

مسئله‌ی ۳.۶. [۱] منظور از یک تور اویلری، گشتی بسته است که از یک رأس شروع می‌شود، همه‌ی یال‌های گراف را دقیقاً یک بار طی می‌کند، و به رأس آغازین برمی‌گردد، و ممکن است از یک رأس چند بار بگذرد. ثابت کنید یک گراف دارای تور اویلری است اگر و تنها اگر همبند باشد و درجه‌ی تمام رئوسش زوج باشد، و این تور را در صورت وجود در $O(V+E)$ بیابید.

ایده‌ی حل: برای یافتن تور اویلری از یک دور شروع کنید و دورهای جدیدی به آن بچسبانید. می‌توانید از لیست پیوندی کمک بگیرید.

۷ زیردرخت فراگیر کمینه

مسئله‌ی ۱.۷. [۲] گرافی با وزن‌های مثبت و متمایز روی یال‌ها داریم. در $O(V^۲)$ آن‌را پیش‌پردازش کنید به طوری که برای هر یال e بتوانید در $O(V)$ زیردرخت فراگیر کمینه‌ای که از این یال می‌گذرد را بیابید.

ایده‌ی حل: زیردرخت فراگیر کمینه بدون محدودیت را محاسبه کنید و آن‌را T بنامید. در $O(V^۲)$، برای هر دو رأس u و v سنگین‌ترین یال روی مسیری که u و v را روی T به هم وصل می‌کند تعیین کنید. حال هر یال $e=uv$ که دادند، جواب مسئله از حذف یال سنگین بین u و v و اضافه کردن e به دست می‌آید.

مسئله‌ی ۲.۷. [۱] گراف G و یک زیردرخت فراگیر کمینه‌ی T از آن داده شده‌اند. یک رأس جدید به گراف اضافه کرده و آن‌را به تعدادی از رئوس وصل می‌کنیم. الگوریتمی بدهید که زیردرخت فراگیر کمینه گراف جدید را به سرعت پیدا کند.

ایده‌ی حل: نکته‌ی مهم این است که زیردرخت فراگیر کمینه‌ای وجود دارد که از یال‌هایی که در G بوده‌اند ولی در T نیستند استفاده نمی‌کند. بنابراین اگر رأس اضافی درجه‌اش d باشد، مسئله به پیدا کردن زیردرخت فراگیر کمینه در گرافی $V+۱$ راسی و $V-۱+d$ یالی تبدیل می‌شود.

۸ کوتاه‌ترین مسیرها در گراف

مسئله‌ی ۱.۸. [۲] در یک گراف جهت‌دار بدون وزن، تعداد کوتاه‌ترین مسیرها از رأس u به v را در $O(V+E)$ بیابید.

ایده‌ی حل: BFS و برای همه‌ی رئوس بین راه تعداد کوتاه‌ترین مسیرها را به دست‌آورید.

مسئله‌ی ۲.۸. [۱] الگوریتمی ارائه دهید که یک دور منفی از گراف داده شده را، در صورت وجود، پیدا کند.

ایده‌ی حل: از الگوریتم Bellman-Ford استفاده کنید.

مسئله‌ی ۳.۸. [۱] اگر وزن هر یک از یال‌های گرافی بتواند یکی از مقادیر $۱, ۲, \ldots, W$ را بگیرد، نشان دهید می‌توان الگوریتم دایکسترا را طوری پیاده‌سازی کرد که مرتبه‌ی زمانی‌اش $O(VW+E)$ باشد.

ایده‌ی حل: طول همه‌ی کوتاه‌ترین مسیرها در گراف عددی بین ۱ و $(V-۱)\times W$ خواهد بود. درنتیجه می‌توانیم آرایه‌ای VW تایی بگیریم و در درآیه‌ی i اُم لیست رئوسی مثل v را قرار دهیم که $d[v]=i$. درنتیجه همه‌ی عملیات «یافتن نزدیک‌ترین رأس»ها را می‌توان به جای Extract-Min از روی پشته، با حرکت روی این آرایه و در زمان $O(VW)$ انجام داد.

مسئله‌ی ۴.۸. [۱] اگر الگوریتمی برای یافتن بستار تراگذری یک DAG با مرتبه‌ی زمانی $f(V,E)$ داشته باشیم، برای گرافی دلخواه الگوریتمی با مرتبه‌ی زمانی $f(V,E)+O(V+E^*)$ ارائه دهید که بستار تراگذری را بیابد. E^* تعداد یال‌های گراف بستار است.

ایده‌ی حل: ابتدا مولفه‌های قویّاً همبند را بسازید، الگوریتم مفروض را روی گراف مولفه‌های قویّاً همبندی اجرا کنید سپس با توجه به نتیجه، بستار تراگذری گراف اوّلیه را بسازید.

۹ شبکه‌ی شارّه

مسئله‌ی ۱.۹. [۱] نشان دهید هر شارّه‌ی بیشینه را می‌توان با augment کردن حداکثر E مسیر افزایشی ساخت.

ایده‌ی حل: فرض کنید شارّه‌ی بیشینه را داریم. یالی با شارّه‌ی مینیمم را درنظربگیرید، و یک مسیر پیدا کنید که از s به t باشد و از آن یال بگذرد. حال به اندازه‌ی شارّه‌ی آن یال از ظرفیت و شارّه‌ی یال‌های این مسیر کم کنید و استقرا بزنید.

مسئله‌ی ۲.۹. [۱] ثابت کنید هر گراف دوبخشی که درجه‌ی تمام رئوسش برابر باشد دارای تطابق کامل است.

ایده‌ی حل: راحت می‌توان دید این گراف شرط هال را دارد، سپس از قضیه‌ی شارّه‌ی بیشینه—برش کمینه استفاده کنید.

مسئله‌ی ۳.۹. فرض کنید ماتریسی با عناصر حقیقی دارید که مجموع عناصر هر سطر و مجموع عناصر هر ستون عددی صحیح است. ثابت کنید می‌توان به جای هر درایه‌ی ماتریس، سقف یا کف آن‌را قرار داد که مجموع عناصر هر سطر و ستون همان مقادیر قبلی بماند.

ایده‌ی حل: می‌توان فرض کرد درایه‌ها بین صفر و یک‌اند. حال گراف دوبخشی بسازید، ظرفیت یال‌های متناظر با عناصر ناصفر را ۱ و بقیه را ۰ بگذارید. طبق فرض این شبکه دارای شارّه‌ی بیشینه‌ی صحیح است لذا طبق قضیه ۲۶.۱۱ از [۱] می‌توان شارّه‌ی بیشینه‌ای یافت که در آن شارّه‌ی همه‌ی یال‌ها صحیح باشد. حال به جای عناصری که شارّه‌ی یال متناظرشان ۱ شده، سقفشان و به جای بقیه کف را قرار دهید.

مسئله‌ی ۴.۹. [۱] ثابت کنید گرچه تعداد relabelهای الگوریتم push-relabel در حالت کلی $O(V^2)$ است و مرتبه‌ی زمانی این عمل ممکن است $O(V)$ باشد ولی انجام تمامی این اعمال $O(VE)$ زمان می‌برد.

ایده‌ی حل: توجه کنید که اگر درجه‌ی رأس v برابر d باشد عمل relabel روی این رأس در $O(d)$ انجام می‌شود.

مسئله‌ی ۵.۹. اگر از صفحه‌ی شطرنج یک خانه‌ی سفید و یک خانه‌ی سیاه حذف کنیم، ثابت کنید شکل حاصل را می‌توان با دومینوها پوشاند به طوری که هر خانه توسط دقیقاً یک دومینو پوشیده‌شود.

ایده‌ی حل: از قضیه‌ی شارّه‌ی بیشینه—برش کمینه استفاده کنید.

۱۰ تطابق رشته‌ها

مسئله‌ی ۱.۱۰. [۱] توضیح دهید که چگونه می‌توان الگوریتم Rabin-Karp را به دو بُعد تعمیم داد.

ایده‌ی حل:

مسئله‌ی ۲.۱۰. [۱] دو رشته‌ی A و B به طول n داده شده‌اند. الگوریتمی با زمان $O(n)$ ارائه دهید که تشخیص دهد آیا اندیس k وجود دارد که به ازای هر $1 \leq i \leq n$ داشته‌باشیم: $B_i = A_{(i+k)\%n}$ (یعنی دو رشته از چرخش هم به دست آمده باشند).

ایده‌ی حل: الگوریتم KMP را برای یافتن الگوی B در رشته‌ی AA به کار برید.

۱۱ پیچیدگی الگوریتم‌ها

مسئله‌ی ۱.۱۱. [۱] الگوریتمی از $O(n^2)$ برای حل مسئله 2-SAT ارائه دهید.

ایده‌ی حل: یک متغیر را مقدار بدهید و ببینید چه متغیرهایی مقدار می‌گیرند. این کار را ادامه دهید تا دیگر متغیری مقدار نگیرد یا به تناقض بخورید. در حالت اول کار را با بقیه‌ی عبارت ادامه دهید و در حالت دوم متغیر را برعکس کنید. اگر باز به تناقض خوردید مسئله قابل ارضا نیست!

مسئله‌ی ۲.۱۱. [۲] ثابت کنید یافتن اندازه‌ی کوچک‌ترین پوشش رأسی برای گراف‌هایی که درجه‌ی تمام رئوس آن‌ها زوج است نیز ان‌پی—تمام است.

ایده‌ی حل: مسئله پوشش رأسی معمولی را به این مسئله کاهش می‌دهیم. فرض کنید گراف G داده شده است. حال سه رأس جدید به آن اضافه کنید و آن سه رأس را به هم وصل کنید، یکی از این سه رأس را به تمام رئوس درجه فرد گراف G نیز وصل کنید. گراف جدید پوشش رأسی به اندازه‌ی $k+2$ دارد اگر و تنها اگر G پوشش رأسی به اندازه‌ی k داشته‌باشد.

مسئله‌ی ۳.۱۱. [۲] ثابت کنید اگر $P \neq NP$ الگوریتمی با زمان چندجمله‌ای وجود ندارد که یک گراف و دو رأس از آن را بگیرد و تشخیص بدهد آیا یک مسیر همیلتونی وجود دارد که دو سرش آن دو رأس داده شده باشند.

ایده‌ی حل: با داشتن چنین الگوریتمی می‌توان مسئله‌ی وجود دور همیلتونی یک گراف دلخواه را در زمان چندجمله‌ای حل کرد.

مسئله‌ی ۴.۱۱. [۳] مسئله‌ی Integer Programming بدین شرح است: آیا اعداد صحیح $a_1, a_2, \ldots, a_n$ وجود دارند که در تعدادی شرط مانند

$$k_1 \times a_1 + k_2 \times a_2 + \ldots + k_n \times a_n > k \qquad k, k_1 \ldots, k_n \in Z$$

صدق کنند؟ ثابت کنید این مسئله ان‌پی—تمام است.

ایده‌ی حل: می‌توان مسئله‌ی subset-sum یا 3-SAT را به این مسئله کاهش داد.

مسئله‌ی ۵.۱۱. [۲] ثابت کنید این مسئله که آیا گرافی دارای رنگ‌آمیزی با ۳ رنگ هست یا خیر، ان‌پی—تمام است.

ایده‌ی حل: مسئله‌ی 3-SAT را می‌توان به این مسئله کاهش داد ولی سخت است.

مسئله‌ی ۶.۱۱. [۱] مسئله‌ی bin-packing بدین شرح است: تعدادی شیء با حجم‌های معلوم را می‌خواهیم در تعدادی جعبه با حجم‌های مساوی قرار دهیم. آیا می‌توان تمام این اشیا را در k تا از این جعبه‌ها قرار داد؟ با کاهش دادن مسئله‌ی subset-sum نشان دهید این مسئله ان‌پی—تمام است.

ایده‌ی حل: فرض کنید یک نمونه (instance) از مسئله‌ی subset-sum به ما داده شده‌است که در آن S مجموع کل اعداد است و باید بگوییم آیا زیرمجموعه‌ای با مجموع k وجود دارد یا خیر. مجموعه‌ای از اشیا بدین شکل می‌سازیم که برای هر عدد یک شیء به آن حجم داریم و دو شیء اضافه نیز با حجم‌های $S+k$ و $2S-k$ داریم. در این صورت مجموع وزن اشیا برابر $S+S+k+2S-k=4S$ می‌باشد. دو جعبه با ظرفیت $2S$ نیز درنظرمی‌گیریم و مسئله‌ی bin packing را حل می‌کنیم.

اگر بتوان این اشیا را در این دو جعبه قرار داد، حتماً دو شیء اضافه شده در دو جعبه‌ی متفاوت قرار دارند، زیرا مجموع وزن آن‌ها $3S$ است که بیشتر از ظرفیت یک جعبه است. به علاوه چون مجموع حجم اشیا دقیقاً برابر مجموع ظرفیت جعبه‌هاست، هر دو جعبه باید پر شده‌باشند. در نتیجه در آن جعبه‌ای که شیء به حجم $2S-k$ قرار داده‌شده، اشیائی قرار گرفته‌اند که مجموع وزنشان $2S-(2S-k)=k$ است. یعنی در واقع یک زیرمجموعه از اعداد پیدا کردیم که جمعشان k شده‌است.

برعکس، اگر یک زیرمجموعه از اعداد وجود داشته‌باشد که جمعشان k شود، اشیای متناظر آن‌ها به علاوه‌ی شیء اضافی به حجم $2S-k$ را در یک جعبه و اشیای مانده را در جعبه‌ی دیگر قرار می‌دهیم، پس مسئله bin packing متناظر جواب دارد.

مسئله‌ی ۷.۱۱. مجموعه‌ی A از رئوس را در گراف G یک مجموعه‌ی قوی می‌نامیم اگر به ازای هر رأسی مثل v از G، یا v در A باشد یا همسایه‌ای در A داشته‌باشد. اگر H یک گراف دوبخشی جهت‌دار با دو بخش X و Y باشد به طوری که جهت همه‌ی یال‌ها از X به Y باشد، زیرمجموعه‌ی S از X را یک مجموعه‌ی خوب می‌نامیم هرگاه به ازای هر رأس v از Y، یالی از یکی از اعضای S به v وجود داشته‌باشد. ثابت کنید:

۱این مسئله که آیا یک گراف مجموعه‌ای قوی به اندازه‌ی k دارد، ان‌پی—تمام است.

۲این مسئله که آیا یک گراف دوبخشی جهت‌دار مجموعه‌ای خوب به اندازه‌ی k دارد، ان‌پی—تمام است.

ایده‌ی حل: حل ۱) مسئله‌ی پوشش رأسی را به مسئله مجموعه‌ی قوی کاهش می‌دهیم. در گراف داده‌شده‌ی G، برای هر یال $e = uv$ یک رأس مثل x_{uv} اضافه کرده و آن‌را به u و v وصل کنید. گراف حاصل که $O(n^۲)$ رأس و یال دارد را H می‌نامیم. ادعا می‌کنیم گراف H یک مجموعه‌ی قوی k رأسی دارد اگر و تنها اگر گراف اولیه G یک پوشش رأسی k رأسی داشته‌باشد.

یک مجموعه‌ی قوی k رأسی برای H درنظربگیرید. اگر این مجموعه شامل رأسی مانند x_{uv} بود، آن‌را حذف کرده و یکی از دو رأس متصل به آن (مثلاً رأس u) را به مجموعه اضافه می‌کنیم. به وضوح مجموعه‌ی جدید نیز قوی است و حداکثر k رأس دارد. از طرفی به راحتی می‌توان دید که این مجموعه از رئوس، یک پوشش رأسی برای G می‌سازند.

برعکس، هر پوشش رأسی برای G دقیقاً یک مجموعه‌ی قوی برای H می‌باشد.

حل ۲) این مسئله دو راه‌حل دارد: می‌توان مسئله‌ی مجموعه‌ی قوی و یا مسئله‌ی پوشش رأسی را به مسئله‌ی مجموعه‌ی خوب کاهش داد.

راه اول (کاهش مسئله‌ی مجموعه‌ی قوی): گراف G با مجموعه رئوس $u_۱, \ldots, u_n$ را درنظر بگیرید، گراف دوبخشی H را بسازید که $X = \{v_۱, \ldots, v_n\}$ و $Y = \{w_۱, \ldots, w_n\}$ دو بخش آن هستند و $v_i \to w_j$ اگر $i = j$ یا u_j و u_i در G مجاور باشند. در این صورت هر مجموعه‌ی قوی $\{u_{i_۱}, \ldots, u_{i_k}\}$ در G دقیقاً متناظر با مجموعه‌ی خوب $\{v_{i_۱}, \ldots, v_{i_k}\}$ در H است و بالعکس.

راه دوم (کاهش مسئله‌ی پوشش راسی): گراف G با مجموعه رئوس $u_۱, \ldots, u_n$ را درنظرگرفته، گراف H را با دو بخش $X = \{v_۱, \ldots, v_n\}$ و Y بسازید که به ازای هر یال $e = uv$ در G، راسی مثل y_{uv} در Y قرار می‌دهیم و $u, v \to y_{uv}$. در این صورت هر پوشش رأسی $\{u_{i_۱}, \ldots, u_{i_k}\}$ در G دقیقاً متناظر با مجموعه‌ی خوب $\{v_{i_۱}, \ldots, v_{i_k}\}$ در H است و بالعکس.

[1] Thomas H. Coremen et al., *Introduction to Algorithms (2nd ed.)*, MIT Press 2001.

[2] Udi Manber, *Introduction to Algorithms: A Creative Approach*, Addison-Wesley 1989.

[3] Steven S. Skiena, *The Algorithm Design Manual*, Springer-Verlag 1998