HW4, Algorithms

Sepehr Ebadi

9933243

Khordad, 1403

سوال ۱:

باید الگوریتمی پیدا کنیم که نفر اول بتواند بیشترین مقدار ممکن را در حالی که جمع ارزش سکه‌های او از K بیشتر نشود. برای این کار، ابتدا باید استراتژی‌ای پیدا کنیم که در هر مرحله بهترین انتخاب را برای نفر اول ارائه دهد:

ابتدا، تمامی سکه‌ها را با ارزش‌هایشان مرتب می‌کنیم.

نفر اول سکه‌ای را انتخاب می‌کند که بیشترین ارزش را داشته باشد و مجموع ارزش‌های او از K بیشتر نشود.

نفر دوم سکه‌های قبلی و بعدی سکه انتخاب شده توسط نفر اول را بردارد.

مراحل ۲ و ۳ را تکرار می‌کنیم تا زمانی که نفر اول نتواند سکه‌ای بیشتر از K بردارد.

الگوریتم:

مرتب کردن لیست سکه‌ها به صورت نزولی براساس ارزش آنها.

ایجاد دو لیست برای نگهداری ارزش سکه‌های برداشته شده توسط نفر اول و نفر دوم.

پیمایش سکه‌های مرتب شده و انتخاب سکه‌های مناسب برای نفر اول تا زمانی که مجموع ارزش‌ها از K بیشتر نشود.

به‌روزرسانی لیست‌های سکه‌های نفر اول و نفر دوم و حذف سکه‌های قبلی و بعدی سکه انتخاب شده توسط نفر اول.

تکرار مراحل بالا تا زمانی که نتوان سکه بیشتری برای نفر اول انتخاب کرد.

ساخت جفت‌های اندیس-ارزش و مرتب‌سازی نزولی:

```
indexed_coins = [(4, 11), (3, 8), (1, 7), (5, 6), (7, 5), (0, 4), (2, 2), (6, 1)]
```

این جفت‌ها بر اساس ارزش سکه‌ها مرتب شده‌اند. برای هر جفت، اولین عدد اندیس و دومین عدد ارزش سکه است.

```
first_player = []
```

second_player = set ()

total_value = 0

پیمایش و انتخاب سکه:

جفت اول:(4, 11)

اندیس ۴ و ارزش ۱۱. نفر اول می‌تواند این سکه را بردارد زیرا مجموع ارزش‌ها از K بیشتر نمی‌شود.

first_player = [4]

total_value = 11

second_player = {3, 5}

جفت دوم:(3, 8)

اندیس ۳ در مجموعه نفر دوم است، پس نفر اول نمی‌تواند این سکه را بردارد.

جفت سوم:(1, 7)

اندیس ۱ و ارزش ۷. نفر اول می‌تواند این سکه را بردارد زیرا مجموع ارزش‌ها 11+7=18 هنوز از K بیشتر نمی‌شود.

first_player = [4, 1]

total_value = 18

second_player = {0, 2, 3, 5}

جفت چهارم:(5, 6)

اندیس ۵ در مجموعه نفر دوم است، پس نفر اول نمی‌تواند این سکه را بردارد.

جفت پنجم:(7, 5)

اندیس ۷ و ارزش ۵. نفر اول می‌تواند این سکه را بردارد زیرا مجموع ارزش‌ها 18+5=23 هنوز از K بیشتر نمی‌شود.

first_player = [4, 1, 7]

total_value = 23

second_player = {0, 2, 3, 5, 6}

باقی جفت‌ها:

اندیس‌های باقی‌مانده (۰، ۲، ۶) همگی در مجموعه نفر دوم هستند، پس نفر اول نمی‌تواند آنها را بردارد.

مجموع ارزش سکه‌های نفر اول: ۲۳

سکه‌های برداشته شده توسط نفر اول: [۱۱, ۷, ۵]

در نتیجه، نفر اول توانسته است سکه‌هایی با ارزش‌های ۱۱، ۷ و ۵ را بردارد که مجموع ارزش آنها ۲۳ است و از محدودیت K=24 تجاوز نکرده است. این استراتژی حریصانه به نفر اول کمک می‌کند تا بیشترین مقدار ممکن را بدون تجاوز از حد مجاز جمع کند.

سوال ۲:

برای محاسبه حداقل تعداد بیت‌های لازم جهت انتقال یا ذخیره‌سازی متنی که شامل حروف آ، ب، پ، ت، ث و ج با تکرارهای مختلف است، از کدگذاری هافمن استفاده می‌کنیم. کدگذاری هافمن یک روش فشرده‌سازی داده‌ها است که از کدهای با طول متغیر استفاده می‌کند و به حروف با فراوانی بیشتر، کدهای کوتاه‌تر اختصاص می‌دهد.

ابتدا، فراوانی هر حرف را محاسبه می‌کنیم:

- آ: ۸۰۰
- ب: ۱۲۰۰
- پ: ۱۰۰۰
- ت: ۱۸۰۰
- ث: ۱۵۰۰
- ج: ۷۰۰

سپس از الگوریتم هافمن برای ساختن درخت هافمن استفاده می‌کنیم:

ساختن گره‌ها و مرتب‌سازی بر اساس فراوانی:

آ: ۸۰۰, ب: ۱۲۰۰, پ: ۱۰۰۰, ت: ۱۸۰۰, ث: ۱۵۰۰, ج: ۷۰۰

مرتب‌سازی:

ج: ۷۰۰, آ: ۸۰۰, پ: ۱۰۰۰, ب: ۱۲۰۰, ث: ۱۵۰۰, ت: ۱۸۰۰

ایجاد درخت هافمن:

مرحله اول: ادغام دو گره با کمترین فراوانی (ج و آ):

گره جدید: (ج + آ) = ۱۵۰۰

گره‌ها: پ: ۱۰۰۰, ب: ۱۲۰۰, (ج + آ): ۱۵۰۰, ث: ۱۵۰۰, ت: ۱۸۰۰

مرحله دوم: ادغام دو گره با کمترین فراوانی (پ و ب):

گره جدید: (پ + ب) = ۲۲۰۰

گره‌ها: (ج + آ): ۱۵۰۰, ث: ۱۵۰۰, (پ + ب): ۲۲۰۰, ت: ۱۸۰۰

مرحله سوم: ادغام دو گره با کمترین فراوانی (ج + آ و ث):

گره جدید: ((ج + آ) + ث) = ۳۰۰۰

گره‌ها: ت: ۱۸۰۰, (پ + ب): ۲۲۰۰, ((ج + آ) + ث): ۳۰۰۰

مرحله چهارم: ادغام دو گره با کمترین فراوانی (ت و پ + ب):

گره جدید: (ت + (پ + ب)) = ۴۰۰۰

گره‌ها: ((ج + آ) + ث): ۳۰۰۰, (ت + (پ + ب)): ۴۰۰۰

مرحله پنجم: ادغام دو گره باقی‌مانده:

گره جدید: (((ج + آ) + ث) + (ت + (پ + ب))) = ۷۰۰۰

ساخت کدهای هافمن:

حالا که درخت هافمن ساخته شده، کدهای هافمن برای هر حرف به شکل زیر است:

ت: ۰

(پ + ب): ۱

پ: ۱۰

ب: ۱۱

((ج + آ) + ث): ۰۰

ث: ۰۱

(ج + آ): ۰۰۰

ج: ۰۰۰۰

آ: ۰۰۰۱

محاسبه طول کدهای هافمن:

- آ: ۴ بیت
- ب: ۲ بیت
- پ: ۳ بیت
- ت: ۱ بیت
- ث: ۲ بیت
- ج: ۴ بیت

محاسبه تعداد کل بیت‌ها:

= تعداد بیت‌های لازم

۴*۷۰۰ + ۲*۱۵۰۰ + ۱*۱۸۰۰ + ۳*۱۰۰۰ + ۲*۱۲۰۰ + ۴*۸۰۰ = ۳۲۰۰ + ۲۴۰۰ + ۳۰۰۰ + ۱۸۰۰

+ ۳۰۰۰ + ۲۸۰۰ =۱۶۲۰۰

بنابراین، حداقل تعداد بیت‌های لازم جهت انتقال یا ذخیره‌سازی این متن با استفاده از کدگذاری هافمن، ۱۶۲۰۰ بیت است.

سوال ۳:

درخت پوشای کمینه (MST) با استفاده از الگوریتم‌های Prim وKruskal

استفاده از الگوریتمPrim

الگوریتم Prim از یک رأس شروع کرده و به تدریج رأس‌های مجاور را اضافه می‌کند تا درخت پوشای کمینه ساخته شود. در اینجا مراحل اجرای الگوریتم Prim را توضیح می‌دهیم:

۱. شروع از رأس ۰:

انتخاب رأس ۰ به عنوان شروع.

کم‌ترین وزن یال‌های مجاور رأس ۰ : (1,0) با وزن ۵ و (3,0)با وزن ۴ و (2,0) با وزن ۷.

انتخاب یال (3,0) با وزن ۴ (کمترین وزن).

۲. اضافه کردن رأس ۳:

یال‌های مجاور رأس‌های ۰ و ۳ : (1,0)با وزن ۵، (2,0)با وزن ۷، (1,3)با وزن ۳، ,(4,3)با وزن ۸، (7,3)با وزن ۲ و (5,3)با وزن ۵.

انتخاب یال (7,3)با وزن ۲ (کمترین وزن).

۳. اضافه کردن رأس ۷:

یال‌های مجاور رأس‌های ۰، ۳ و ۷ : (1,0)با وزن ۵، (2,0)با وزن ۷، (1,3)با وزن ۳، (4,3)با وزن ۸، (4,7)با وزن ۱، (6,7)با وزن ۷ و (5,7) با وزن ۹.

انتخاب یال (4,7)با وزن ۱ (کمترین وزن).

۴. اضافه کردن رأس ٤:

یال‌های مجاور رأس‌های ۰، ۳، ۷ و ۴ : (1,0)با وزن ۵، (2,0)با وزن ۷، (1,3)با وزن ۳، (1,4) با وزن ۶، (6,4)با وزن ۴ و (6,7)با وزن ۷.

انتخاب یال (6,4)با وزن ۴ (کمترین وزن).

۵. اضافه کردن رأس ٦:

یال‌های مجاور رأس‌های ۰، ۳، ۷، ۴ و ۶ : (1,0)با وزن ۵، (2,0)با وزن ۷، (1,3)با وزن ۳، (1,4)با وزن ۶ و (5,3)با وزن ۵.

انتخاب یال (1,3)با وزن ۳ (کمترین وزن).

۶. اضافه کردن رأس ۱:

یال‌های مجاور رأس‌های ۰، ۳، ۷، ۴، ۶ و ۱: (1,0) با وزن ۵ و (2,1)با وزن ۱.

انتخاب یال (2,1)با وزن ۱ (کمترین وزن).

۷. اضافه کردن رأس ۲:

یال‌های مجاور رأس‌های ۰، ۳، ۷، ۴، ۶، ۱ و ۲ : (1,0) با وزن ۵ و (5,3)با وزن ۵.

انتخاب یال (5,3)با وزن ۵ (کمترین وزن).

۸. اضافه کردن رأس ۵:

تمامی رأس‌ها به درخت پوشای کمینه اضافه شدند.

بنابراین، MSTبا استفاده از الگوریتم Prim:

یال‌های انتخاب شده :

(3,0), (7,3), (4,7), (6,4), (1,3), (2,1), (5,3)

- مجموع وزن‌ها

4+2+1+4+3+1+5 = 20

استفاده از الگوریتمKruskal

الگوریتم Kruskal یال‌ها را به ترتیب صعودی از نظر وزن انتخاب می‌کند و آنها را اضافه می‌کند تا زمانی که یک درخت پوشای کمینه ساخته شود.

۱. مرتب کردن یال‌ها به ترتیب صعودی وزن:

- (2,1)با وزن ۱
- (4,7)با وزن ۱
- (7,3)با وزن ۲
- (1,3)با وزن ۳
- (3,0)با وزن ۴
- (6,4)با وزن ۴
- (5,3)با وزن ۵
- (1,0)با وزن ۵
- (5,2)با وزن ۲
- (2,0)با وزن ۷
- (6,7)با وزن ۷
- (1,4)با وزن ۶
- (4,3)با وزن ۸
- (5,7)با وزن ۹

۲. **انتخاب یال‌ها به ترتیب وزن، بدون ایجاد چرخه:**

- (2,1)با وزن ۱
- (4,7)با وزن ۱

- (7,3)با وزن ۲
- (1,3)با وزن ۳
- (3,0)با وزن ۴
- (6,4)با وزن ۴
- (5,3)با وزن ۵

بنابراین، MSTبا استفاده از الگوریتم Kruskal:

- یال‌های انتخاب شده :

(2,1), (4,7), (7,3), (1,3), (3,0), (6,4), (5,3)

- مجموع وزن‌ها :

1+1+2+3+4+4+5=20

مقایسه پیچیدگی زمانی

- **الگوریتم Prim :** پیچیدگی زمانی $O(V^2)$ با استفاده از ماتریس مجاورت و O(E+VlogV) با استفاده از صف اولویت.

الگوریتم Kruskal :پیچیدگی زمانی O(ElogE+ElogV) با استفاده از ساختار داده‌ی مجموعه مجزا.

در عمل، الگوریتم Prim معمولاً برای گراف‌های چگال مناسب‌تر است، در حالی که الگوریتم Kruskalبرای گراف‌های تنک مناسب‌تر است.

سوال ٤:

برای حل مسئله‌ی حداقل تعداد پخش اطلاعیه به طوری که همه‌ی کلاس‌ها حداقل یک بار آن را بشنوند، می‌توانیم از یک الگوریتم حریصانه استفاده کنیم. الگوریتم حریصانه برای حل این مسئله بهینه است و به مسئله‌ی پوشش بازه‌ها (Interval Covering) معروف است.

الگوریتم حریصانه(Greedy Algorithm)

۱. **مرتب‌سازی بازه‌ها بر اساس زمان پایان :**ابتدا تمامی بازه‌ها (زمان‌های برگزاری کلاس‌ها) را بر اساس زمان پایانشان به صورت صعودی مرتب می‌کنیم.

۲. **انتخاب بازه‌ها :**سپس بازه‌ها را یکی یکی انتخاب می‌کنیم. برای هر بازه انتخاب شده، یک نقطه‌ی اطلاع‌رسانی در زمان پایان آن بازه قرار می‌دهیم. بازه‌های بعدی که شامل این نقطه می‌شوند را نادیده می‌گیریم، چرا که اطلاعیه در این زمان برای آن‌ها نیز کافی است.

اثبات بهینگی الگوریتم حریصانه

این الگوریتم به این دلیل بهینه است که در هر مرحله، بازه‌ای را انتخاب می‌کند که زمان پایان آن کمترین باشد. به این ترتیب، بیشترین تعداد بازه‌ها را با یک اطلاعیه پوشش می‌دهد.

مثال و توضیح با شکل داده شده

در شکل داده شده، بازه‌های زمانی کلاس‌ها به صورت افقی نمایش داده شده‌اند. فرض کنید بازه‌های زمانی به صورت زیر باشند (با شروع و پایان):

کلاس ۱: (4, 1)

کلاس ۲: (5, 2)

کلاس ۳: (6, 3)

کلاس ۴: (7, 4)

کلاس ۵ : (8, 5)

کلاس ۶: (9, 6)

کلاس ۷: (10, 7)

کلاس ۸: (11, 8)

کلاس ۹: (12, 9)

کلاس ۱۰: (13, 10)

کلاس ۱۱ : (14, 11)

کلاس ۱۲: (15, 12)

مراحل الگوریتم حریصانه

۱. **مرتب‌سازی بر اساس زمان پایان :**کلاس‌ها را بر اساس زمان پایانشان مرتب می‌کنیم:

(1, 4), (2, 5), (3, 6), (4, 7), (5, 8), (6, 9), (7, 10), (8, 11), (9, 12), (10, 13), (11, 14), (12, 15)

۱. **انتخاب بازه‌ها:**

- انتخاب اولین بازه با زمان پایان ۴. اطلاع‌رسانی در زمان ۴ انجام می‌شود.
- بازه‌های پوشش داده شده: (۱, ۴)، (۲, ۵)،(۳,۶)
- انتخاب بازه بعدی با زمان پایان ۷. اطلاع‌رسانی در زمان ۷ انجام می‌شود.
- بازه‌های پوشش داده شده: (۴, ۷)، (۵, ۸)،(۶,۹)
- انتخاب بازه بعدی با زمان پایان ۱۰. اطلاع‌رسانی در زمان ۱۰ انجام می‌شود.
- بازه‌های پوشش داده شده: (۷, ۱۰)، (۸, ۱۱)، (۹,۱۲)
- انتخاب بازه بعدی با زمان پایان ۱۳. اطلاع‌رسانی در زمان ۱۳ انجام می‌شود.

- بازه‌های پوشش داده شده: (۱۰, ۱۳)، (۱۴,۱۱)
- انتخاب بازه بعدی با زمان پایان ۱۵. اطلاع‌رسانی در زمان ۱۵ انجام می‌شود.
- بازه‌های پوشش داده شده (۱۵,۱۲).

بنابراین، تعداد اطلاع‌رسانی‌های مورد نیاز ۵ دفعه است.