الگوریتم‌های پیشرفته

پاسخ تمرین سری ششم

---

1) مرتبه‌ی زمانی الگوریتم تقریبی ارائه شده $O(n)$ است. این الگوریتم یک الگوریتم 2-تقریب است. اثبات: دو جعبه‌ی متوالی را در نظر بگیرید. مجموع وزن وسایل موجود در این دو جعبه باید بیشتر از c باشد. زیرا اگر مجموع وسایل موجود در این دو جعبه کمتر از c بود، این الگوریتم همه‌ی وسایل موجود در جعبه‌ی دوم را در جعبه‌ی اول میگذاشت. این مسئله برای هر دو جعبه‌ی متوالی صحیح است. بنابراین در صورت استفاده از این الگوریتم، حداکثر $\frac{1}{2}$ از ظرفیت وزنی خالی می‌ماند. بنابراین تعداد جعبه‌های لازم در این الگوریتم حداکثر دو برابر تعداد جعبه‌های لازم در الگوریتم بهینه است.

2)

الف) یک راه حل greedy برای این مسئله به این صورت است که پردازش‌ها به صورتی sort شوند که $p_1 \geq p_2 \geq \ldots \geq p_m$ باشد. با در نظر گرفتن این ترتیب (یعنی با شروع از پردازش با زمان p_1)، در هر مرحله یکی از پردازش‌ها به پردازنده‌ای اختصاص داده می‌شود که مدت زمانی که تا این لحظه فعالیت داشته نسبت به سایر پردازنده‌ها کمتر باشد.

ب) برای این مثال، ابتدا پردازش‌ها به صورت {6,6,4,4,3,1} مرتب می‌شوند. سپس اگر از الگوریتم قسمت (الف) استفاده شود، مدت زمان فعالیت پردازنده‌ها به صورت زیر خواهد بود:

پردازنده‌ی 1:

$$T_1 = p_1 + p_3 + p_5 = 6 + 4 + 3 = 13$$

پردازنده‌ی 2:

$$T_2 = p_2 + p_4 = 6 + 4 + 1 = 11$$

بنابراین در این حالت، $M = 13$ است. یک زمانبندی بهینه برای این مثال می‌تواند به صورت زیر باشد:

پردازنده‌ی 1:

$$T_1 = 6 + 6 = 12$$

پردازنده‌ی 2:

$$T_2 = 4 + 4 + 3 + 1 = 12$$

بنابراین پاسخ بهینه برای این مثال $M = 12$ است و تقریب الگوریتم greedy قسمت (الف) برای این مثال $\frac{13}{12}$ می‌باشد.

ج) با فرض آنکه M^* پاسخ بهینه‌ی این مسئله باشد، $M^* \geq max\, p_i$ است و $M^* \geq \frac{1}{n}\sum_{i=1}^{m} p_i$ می‌باشد. بنابراین می‌توان نتیجه گرفت:

$$M^* \geq \frac{1}{2}\left(max\, p_i + \frac{1}{n}\sum_{i=1}^{m} p_i\right)$$

فرض می‌شود T_k پاسخ روش greedy برای این مسئله باشد (پردازنده‌ی k دارای بیشترین مدت زمان فعالیت است). فرض می‌شود پردازش j، آخرین پردازشی باشد که روی پردازنده‌ی k اجرا می‌شود. بنابراین $p_j \leq max\, p_i$ است. همچنین با توجه به الگوریتم greedy، بلافاصله قبل از شروع اجرای پردازش j، پردازنده‌ی k کمترین مدت زمان فعالیت را در بین پردازنده‌ها داشته است. بنابراین:

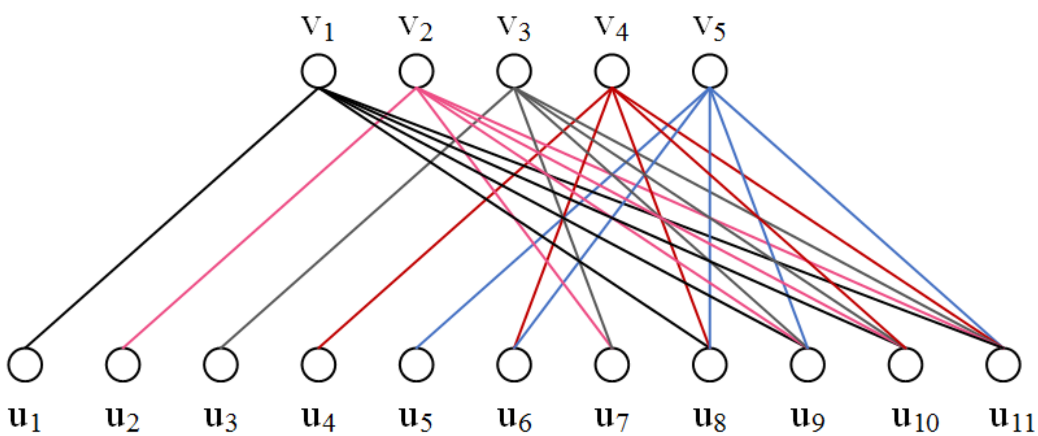
$$T_k - p_j \leq \frac{1}{n}\sum_{i=1}^{m} p_i$$

بنابراین:

$$T_k = p_j + (T_k - p_j) \leq max\, p_i \; + \frac{1}{n}\sum_{i=1}^{m} p_i$$

چنانکه گفته شد، $M^* \geq \frac{1}{2}\left(max\, p_i \; + \frac{1}{n}\sum_{i=1}^{m} p_i\right)$ است. بنابراین الگوریتم greedy ارائه شده در قسمت (الف) یک الگوریتم α-تقریب است و $\alpha \leq 2$ می‌باشد. (همچنین می‌توان اثبات کرد که $\alpha \leq \frac{3}{2}$ است).

3) برای حل این سوال می‌توان یک گراف دو بخشی مانند شکل زیر را در نظر گرفت:



مجموعه‌ی گره‌های این گراف به صورت $V = A \cup B$ است به طوری که $A = \{v_1, v_2, ..., v_5\}$ و $B = \{u_1, u_2, ..., u_{11}\}$ می‌باشند. در این مثال، درجه‌ی همه‌ی گره‌های موجود در A عدد 5 است و مجموعه‌ی A یک پوشش رأسی برای این گراف است. اما ممکن است با استفاده از روش ابتکاری ارائه شده در صورت مسئله، مجموعه‌ی B به عنوان پوشش رأسی گراف تعیین شود: به این صورت که ابتدا گره‌ی u_{11} که درجه‌ی آن 5 است، به مجموعه‌ی پوشش رأسی اضافه می‌شود و همه‌ی یال‌های مجاور آن حذف می‌شوند. در گراف حاصل، درجه‌ی همه‌ی گره‌ها کوچکتر یا مساوی 4 است. بنابراین در این مرحله u_{10} می‌تواند به پوشش رأسی اضافه شود و یال‌های مجاور آن حذف می‌شوند. سپس u_9 به مجموعه‌ی پوشش رأسی اضافه می‌شود و این روند

ادامه پیدا می‌کند تا زمانی که همه‌ی گره‌های موجود در مجموعه‌ی B، در مجموعه‌ی پوشش رأسی اضافه شده باشند. بنابراین ترتیب اضافه شدن گره‌ها به مجموعه‌ی پوشش رأسی می‌تواند به صورت دنباله‌ی زیر نشان داده شود:

$$\langle u_{11}, u_{10}, u_9, u_8, u_7, u_6, u_5, u_4, u_3, u_2, u_1 \rangle$$

در این حالت، $|A| = 5$ و $|B| = 11$ است، بنابراین $|B| > 2|A|$ می‌باشد.

4) روش ارائه شده در این سوال، یک الگوریتم 2-تقریب است. الگوریتم پریم[1] برای پیدا کردن درخت پوشای کمینه، در هر مرحله نزدیک‌ترین گره به گره‌هایی که تا آن مرحله در نظر گرفته شده‌اند را پیدا می‌کند و آن را به درخت اضافه می‌کند و محل گره‌ی جدید در درخت را مجاور نزدیکترین گره‌ای که قبلا در درخت موجود بوده، در نظر می‌گیرد. با استفاده از اضافه کردن گره‌ها به دور با این روش، فرزند هر گره از درخت همواره پس از آن گره در نظر گرفته می‌شود. در این روش در هر مرحله، دور ایجاد شده یک پیمایش preorder از درخت پوشای کمینه می‌باشد که تا آن مرحله ایجاد شده است؛ در انتهای الگوریتم نیز این ویژگی وجود دارد. یعنی زمانی که همه‌ی گره‌ها به دور اضافه شدند، یک پیمایش preorder از درخت پوشای کمینه ایجاد شده است. چنانچه در فصل 35-2 کتاب CLRS توضیح داده شده است، چنین دور همیلتونی تقریب 2 دارد (با فرض برقراری شرایط نامساوی مثلث).

5) ابتدا بیشترین مقدار ظرفیت فلَش‌های موجود تعیین می‌شود (یعنی بیشترین مقدار در آرایه‌ی C) و این عدد در متغیر m ذخیره می‌شود. سپس $d = \frac{m \times \varepsilon}{k}$ محاسبه می‌شود. همه‌ی مقادیر موجود در C به d تقسیم می‌شوند و حاصل آن در آرایه‌ی C' ذخیره می‌شود. بنابراین:

$$C^{'}[i] = \lfloor \frac{C[i]}{d} \rfloor$$

[1] Prim's algorithm

الگوریتم برنامه‌نویسی پویا برای مقادیر موجود در C' اجرا می‌شود (سایر پارامترها مانند حالت قبل باقی می‌مانند). مرتبه‌ی زمانی این الگوریتم به صورت خطی نسبت به k و ε می‌باشد. مقدار تقریب این الگوریتم $1 - \varepsilon$ است. در این روش تعدادی از رقم‌ها که اهمیت کمتری دارند، گرد می‌شوند. اثبات مقدار تقریب این الگوریتم:

فرض می‌شود که A مجموعه‌ی فلَش‌هایی باشد که توسط این الگوریتم تقریبی تولید می‌شود و $P(A)$ مجموع ظرفیت فلَش‌های موجود در A باشد. همچنین $P'(A)$ مجموع ظرفیت فلَش‌های موجود در A با توجه به مقادیر موجود در آرایه‌ی C' است. همچنین فرض می‌شود که O مجموعه‌ی تولید شده توسط پاسخ بهینه باشد. در این حالت هدف آن است گزاره‌ی زیر اثبات شود:

$$P(A) \geq (1 - \varepsilon) \times P(O)$$

با توجه به اینکه $C^{'}[i] = \lfloor \frac{C[i]}{d} \rfloor$ می‌باشد، می‌توان نتیجه گرفت که:

$$P(O) - d \times P'(O) \leq k \times d$$

پس از اجرای مرحله‌ی برنامه‌نویسی پویا، مجموعه‌ای حاصل می‌شود که با توجه به مقادیر موجود در آرایه‌ی C' بهینه است. بنابراین:

$$P(A) \geq d \times P^{'}(O) \geq P(O) - k \times d = P(O) - m \times \varepsilon$$

همچنین $P(O) \geq m$ می‌باشد. بنابراین:

$$P(A) \geq (1 - \varepsilon) \times P(O)$$

6) برش جدا کننده برای گره‌ی $a_i \in A$، یک مجموعه از یال‌های گراف است که a_i را از سایر گره‌های موجود در A جدا می‌کند. مسئله‌ی پیدا کردن برش جداکننده با کمترین وزن برای گره‌ی $a_1 \in A$، قابل کاهش به مسئله‌ی برش کمینه است. برای این کار، یک گراف جدید ساخته می‌شود که در آن یک گره مانند $a_i^{'}$ به جای همه‌ی گره‌های موجود در $A - \{a_i\}$ گذاشته می‌شود. در این گراف، یال‌های خارج شونده از گره‌ی $a_i^{'}$ متناظر با یال‌های خارج شونده از گره‌های موجود در $A - \{a_i\}$ در گراف اولیه می‌باشند. برش کمینه‌ی

بین دو گره‌ی a_i و $a_i^{'}$ در گراف جدید، نشان دهنده‌ی برش جداکننده با کمترین وزن برای گره‌ی a_i در گراف اولیه است.

الگوریتم تقریبی: ابتدا برش جداکننده با کمترین وزن برای هر $a_i \in A$ محاسبه می‌شود؛ این برش جداکننده D_i می‌باشد. در صورتی که هر $n - 1$ عدد از n برش جداکننده با یکدیگر ادغام شوند، این برش هریک از $n - 1$ عدد a_i را جدا می‌کند. بنابراین این برش، a_i باقی مانده را نیز جدا می‌کند. در این الگوریتم تقریبی، $n - 1$ عدد برش جدا کننده‌ی کم‌وزن‌تر به عنوان پاسخ سوال در نظر گرفته می‌شوند. مقدار تقریب این الگوریتم $2 - \frac{2}{n}$ می‌باشد. اثبات مقدار تقریب الگوریتم:

باید اثبات شود که مجموع وزن $n - 1$ عدد برش جداکننده‌ی کم‌وزن‌تر مجموعه‌ی A، حداکثر $2 - \frac{2}{n}$ برابر وزن پاسخ بهینه است.

فرض می‌شود که OPT، پاسخ بهینه‌ی این مسئله برای مجموعه‌ی A باشد. اگر همه‌ی یال‌های موجود در OPT از گراف G حذف شوند، n مولفه‌ی همبند به صورت $V_1, ..., V_n$ باقی می‌مانند به طوری که V_i شامل a_i است. فرض می‌شود P_i مجموعه‌ی یالهای گراف G باشد که بین V_i و $V - V_i$ هستند به طوری که P_i یک برش جداکننده برای a_i باشد. هر یال $e \in OPT$، بین دو مولفه‌ی مختلف مثلا V_i و V_k قرار دارد به طوری که $e \in P_i$ و $e \in P_k$ می‌باشد. بنابراین هر term در مجموع وزن یال‌ها در $weight(OPT)$، دو بار در مجموع وزن‌های $weight(P_1) + weight(P_2) + ... + weight(P_n)$ وجود دارد. بنابراین:

$$\sum_{i=1}^{n} weight(P_i) = 2weight(OPT)$$

فرض می‌شود که برش P_n، بیشترین وزن را در میان همه‌ی P_i ها داشته باشد. بنابراین:

$$\sum_{i=1}^{n-1} weight(P_i) \leq 2\left(\frac{n-1}{n}\right)weight(OPT) = \left(2 - \frac{2}{n}\right)weight(OPT)$$

چنانکه گفته شد، هر P_i یک برش جداکننده برای a_i می‌باشد و چون D_i، برش جداکننده با کمترین وزن برای a_i است، بنابراین $weight(D_i) \leq weight(P_i)$ می‌باشد (به ازای همه‌ی مقادیر i). بنابراین:

$$\sum_{i=1}^{n-1} wight(D_i) \leq \sum_{i=1}^{n-1} weight(P_i) \leq \left(2 - \frac{2}{n}\right)weight(OPT)$$