دانشگاه صنعتی اصفهان، دانشکده برق و کامپیوتر

طراحی الگوریتم ها – تکلیف سری آخر

پاسخنامه

قسمت اول

---

1. فرض کنید که P!= NP

(A) NP-complete = NP

(B) $NP-Complete \cap P = \emptyset$

(C) NP-hard = NP

(D) P = NP-complete

گزینه ی B زیرا::

با توجه به فرض سؤال داریم P != NP

طبق تعریف NP-complete ها مسائلی هستند که از تمام مسائل NP سخت تر هستند. پس اگر عضوی در NP-complete ها باشد از P سخت تر است و چون P != NP است و P زیر مجموعه ی NP است پس اشتراک NP-complete و P تهی میشود. زیرا اگر تهی نشود یعنی مسأله ای در P وجود دارد که از تمام مسائل NP سخت تر است و تمام مسائل به آن کاهش می‌یابد و این باعث می‌شود که P = NP باشد که بر خلاف فرض سؤال است.

2. فرض کنید S یک مسأله ی NP-complete است و Q و R دو مسأله هستند که NP نیستند. مسأله ی Q را میتوانیم به مسأله ی S کاهش دهیم(در زمان چند جمله ای) و همچنین مسأله ی S را میتوانیم کاهش دهیم به مسأله ی R (در زمان چند جمله ای)

کدام یک از گزینه ها درست است؟

(A) R عضو NP-complete

(B) R عضو NP-hard

(C) Q عضو NP-complete

(D) Q عضو NP-hard

گزینه ی B :

گزینه ی A و گزینه ی C غلط است زیرا طبق فرض سؤال R و Q عضو NP نیستند پس نمیتوانند عضو NP-complete هم باشند.

طبق صورت سؤال مساله ی R از مسأله ی S سخت تر است و چون S یک مسأله ی NP-complete است پس R یک مسأله ی NP-hard میتواند باشد

تعریف : NP-hard دسته‌ای از مسائل هستند که از تمام مسائل NP سخت تر هستند.

تعریف : NP-complete دسته‌ای از مسائل NP هستند که از تمام مسائل NP سخت تر هستند

تعریف: اگر حل مسأله ی A در زمان چند جمله‌ای کاهش یابد به حل مسأله ی B آنگاه مسأله ی B از مسأله ی A سخت تر است.

3. فرض کنید X یک مسأله ی متعلق به NP باشد. کدام یک از گزینه های زیر درست است؟
الف) برای X هیچ الگوریتم چند جمله‌ای وجود ندارد

ب) اگر X بتواند در زمان چند جمله‌ای به شکل deterministic حل شود آنگاه P = NP است

ج) اگر X عضو NP-hard باشد آنگاه X یک مسأله ی NP-complete است

د) X شاید غیرقابل تصمیم گیری باشد(undecidable)

گزینه ی ج

مسائل P نیز زیرمجموعه ی مسائل NP هستند و برای آن‌ها الگوریتم چند جمله‌ای وجود دارد پس گزینه ی الف غلط است. اگر X عضو P باشد پس هم‌اکنون برای آن الگوریتم deterministic وجود دارد و دلیل بر P = NP نمیشود (همانند قسمت الف)

طبق تعریف اگر X عوض NP-hard باشد پس از تمام مسائل NP سخت تر است. همچنین طبق صورت سؤال X عضوی از NP است. پس X در شرایط تعریف NP-complete صدق کرده و دارای شرط complete بودن است.
پس این گزینه درست است

4. در رابطه با مسائل Sat-3 و Sat-2

الف) هر دو در P هستند

ب) هر دو NP-complete هستند

ج) مسأله ی Sat-3 یک مسأله ی NP-complete است و Sat-2 در P است

د) الف و ب

گزینه ی ج

گزینه ی P غلط است زیرا Sat-3 یک مسأله ی P نیست و NP است. به طور دقیق‌تر sat-3 یک مسأله ی NP-complete است.

گزینه ی ب غلط است زیرا برای sat-2 الگوریتم polynomial وجود دارد پس sat-2 نمیتواند NP-complete باشد (مگر در صورت NP=P بودن)

گزینه ی ج درست است (طبق توضیحات قبل)

5. کدام یک از جملات زیر درست است؟؟ (ممکن است بیش از یک جمله درست باشد)

جمله ی اول) مسأله ی وجود دور در یک گراف بدون جهت دار یک مسأله ی P است

جمله ی دوم) مساله ی جمله ی قبلی در یک مسأله ی NP است

جمله ی سوم) اگر یک مسأله NP-complete باشد برای آن یک الگوریتم non-deterministic وجود دارد که در زمان چند جمله‌ای آن مسأله را حل کند

جمله ی اول :: درست است. با استفاده از تغییرات مناسب در الگوریتم dfs میتوان دور را نیز تشخیص داد.

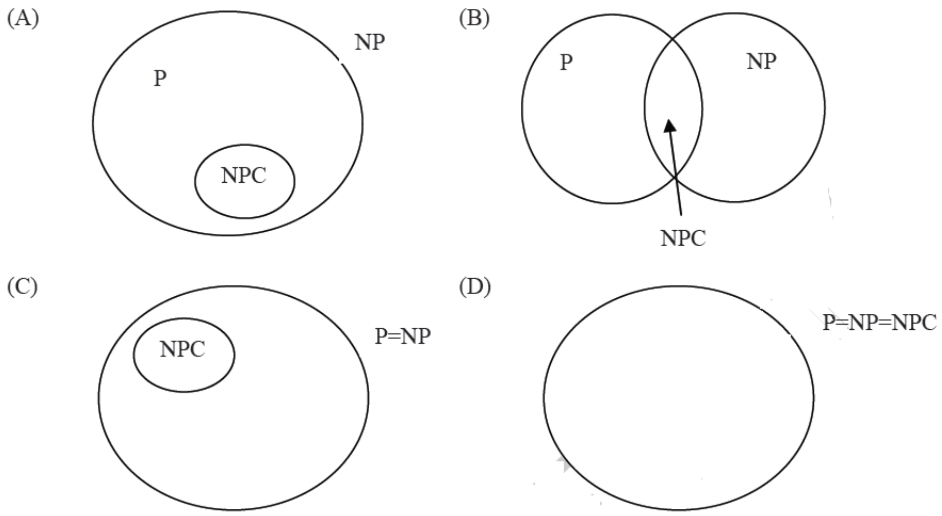
جمله ی دوم :: درست است زیرا P زیرمجموعه ی NP است

تعریف : مسأله ای NP است که بتوان در صورت داشتن یک جواب، درستی جواب را در زمان چند جمله‌ای بررسی کرد.

جمله ی سوم :: درست است زیرا NP-complete زیرمجموعه ای از NP است و مسائل NP-complete همگی در NP هستند. همانطور که از اسم NP پیداست پس برای آن‌ها non-deterministic polynomial time algorithm وجود دارد.

6. فرض کنید یک الگوریتمی کشف شد که مسأله ی بزرگترین clique در یک گراف را حل کند.

کدام یک از نمودار های زیر آن گاه صحیح است؟



اگر چنین الگوریتمی پیدا شد پس اثبات شده است که P=NP است. طبق تعریف NP-complete پس NP=NP-complete نیز میشود. پس شکل D درست است.

قسمت دوم

---

1. در قسمت الگوریتم های حریصانه با مسأله ی رنگ‌آمیزی و الگوریتمی برای رنگ‌آمیزی گراف ها با دو رنگ آشنا شدیم

حال فرض کنید که در همان مسأله میخواهیم گراف را با m رنگ، رنگ‌آمیزی کنیم (برای سادگی کار فرض کنید m = 3 است)

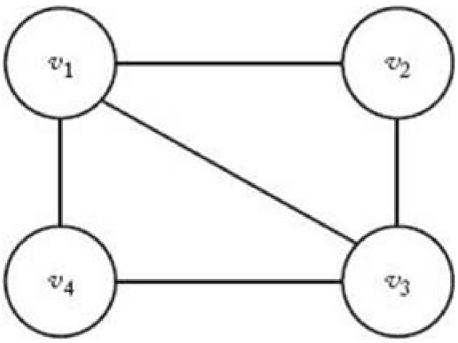
الگوریتمی ارائه دهید که گراف مورد نظر را بررسی کند و بگوید آیا این گراف را میتوان با m رنگ، رنگ‌آمیزی کرد یا خیر

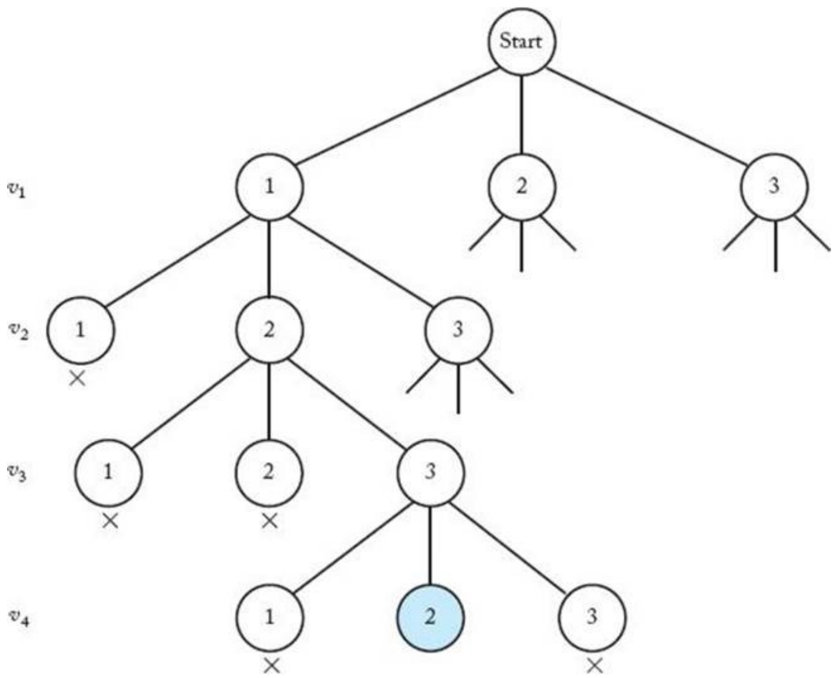
الگوریتم خود را از نظر زمان بررسی کنید (بررسی از نظر worst-case کافی است)

برای حل این مسأله میتوان از الگوریتم های backtracking استفاده کرد. بدین شکل که در سطح اول رنگ گره ی اول را بررسی میکنیم و در سطح دوم رنگ گره ی دوم و در سطح سوم رنگ گره ی سوم و...

در این حالت branching factor برای هر گره نیز m میشود.

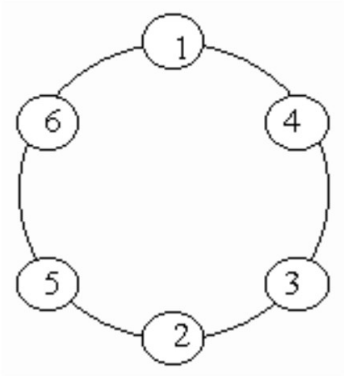
برای مثال برای درخت زیر backtracking زیر را داریم





برای تحلیل زمانی نیز همانطور که بحث شد در بدترین حالت ما مجبور به پیمایش کامل درخت حالت خود هستیم پس در بدترین حالت دارای $O(m^{v+1})$ هستیم.

2. عدد طبیعی n را در نظر بگیرید. اعداد از یک تا n را روی یک دایره کنار هم بچینید به گونه ای که جمع هر دو عدد که کنار یکدیگر واقع شده اند یک عدد فرد بشود. مثلا برای n = 6 به شکل زیر میشود:



الگوریتمی ارائه دهید که همه ی جواب های ممکن برای یک عدد n را بدست آورد.

در این حالت مسلماً بایستی اعداد کنار هم یک عدد زوج و یک عدد فرد باشد. کافی است درخت حالت را رسم کرده و تمام حالت‌ها را بررسی کنیم. یعنی در خانه ی اول میتواند n/2 عدد فرد باشد. در خانه ی بعدی می‌تواند n/2 عدد زوج باشد. در خانه ی بعدی n/2-1 حالت عدد فرد میتواند باشد و به همین صورت

3. مسأله ی امتیازی :: در فصل های قبل با مسئله برنامه ریزی (scheduling problem) آشنا شدیم. مانند قبل در این مسئله نیز یک منبع را برای بازه ای در اختیار داریم و تصمیم داریم بیشترین کارها را در این بازه انجام دهیم، با این تفاوت که در اینجا کارها به زیر مجموعه از بازه ها زمانی نیاز دارد. به عنوان مثال یک کار می تواند به بازه های 10 تا 11 و 14 تا 15 نیاز داشته باشد. حال به شما n فعالیت داده شده است که هر کدام با مجموعه ای از بازه های

زمانی مشخص شده است، و شما باید به این پرسش پاسخ دهید که اگر به ما عدد K را بدهند آیا می‌توانیم حداقل k فعالیت را بپذیریم به شکلی که این فعالیت ها هم پوشانی نداشته باشند؟
راهنمایی: این مسئله را نیز می توانید با کاهش به مسئله independent set حل کنید!