تمرینات سری ۳ (پاسخ)

الگوریتم های حریصانه

چند نکته :

- زمان کلاس رفع اشکال مخصوص امتحان میان ترم :

  شنبه ۹ اردیبهشت، ساعت ۱۲ الی ۱۳:۱۵ (محل کلاس متعاقبا اعلام خواهد شد)

- برای سودمندتر شدن این جلسه، دوستان پیشنهادات خود را برای نحوه برگزاری این جلسه و سوال های مدنظر خود را به بنده (EmadPres@gmail.com) اطلاع دهید.
  - اولویت با حل تمرینات جزوه میباشد.
  - اولویت با حل تمرینات مباحث "تقسیم و حل" و "حریصانه" میباشد .
  - اولویت با حل تمرینات ارسالی میباشد.

- آن دسته از دوستانی که بعضی از تکالیف خود را (به دلایل مختلف :-") ارسال نکرده‌اند، میتوانند از کلاس حل تمرین این هفته بعنوان یک فرصت برای تحویل حضوری تکالیف استفاده نمایند.

- منابع مطالعاتی برای امتحان میان ترم :
  - جزوه اصلی درس
  - کتاب: طراحی الگوریتم (نیپو لیتان) (**Foundations of Algorithms 3rd**)

- کتاب CLRS 3rd

مباحث امتحان میانترم :

سسسسسسسسسسس

پاسخ تمرین ها

ایده سوال یک :

- تا جایگاه سوخت اول برو (*d[0]<m*)
- (در هر جایگاه) آیا با سوخت فعلی به جایگاه بعدی میرسی ؟
- بلی : سوختگیری نکن و تا جایگاه بعدی ادامه بده
- خیر: سوختگیری کن

علت درستی :

اگر لیستی از مکان های سوخت گیری با الگوریتم حریصانه فوق را *L_Gas* بنامیم، فرض میکنیم جواب بهینه دیگری بنام *O_Gas* وجود دارد که در تا جایگاه *i-1* یکسان بوده و در جایگاه *i*ام متفاوتند. این بدین معناست که اختلاف فاصله دو جایگاه *i-1* ام و *i*ام در *O_Gas* از *L_Gas* بیشتر است !! .....

ایده سوال دو :

- در این سوال باید بصورت معکوس فکر کنیم
- یعنی در ابتدا فرض میکنیم که تنها یک چادر (*N=1*) برای پوشش غرفه های خواسته شده(خاکستری رنگ)، به ما داده شده است
- برای کمینه کردن تعداد چادر های کل، کافیست باز هایی با بیشترین تعداد چادر های غیر خاکستری رنگ را حذف کنیم
- یعنی به ازای ورودی *N*، *N-1* مرحله به حذف بازهایی با ویژگی فوق میپردازیم

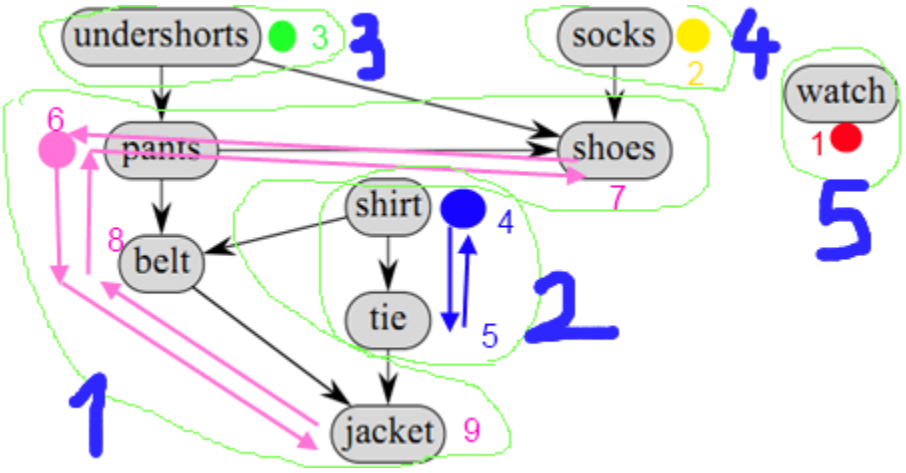
ایده سوال سه:

- در این سوال ما از الگوریتم پیمایش گراف، *DFS* کمک میگیریم
- فرم بازگشتی الگوریتم *DFS* را به یاد بیاورید ...

```
1 Void DFS( _graph, _vertex )
2     for all vertex connected to _vertex ( call them ver ):
3           if ( ver is not visited ) :
4                 DFS( graph, ver );
5     return;
```

- خط ۵ام محل بازگشت الگوریتم از یک راس، پس از پیمودن آن است
- این بدان معناست که اگر ما از کارها، گراف جهت داری (از سمت پیشنیاز به پسنیاز) تشکیل بدهیم، هنگامی که عمل پیمایش از یک راس در حال بازگشت است (همان خط ۵ام کد)، میتوان با خیال راحت فرض کرد، پسنیاز کار فوق قبلا پیمایش شده است !
- در نتیجه اگر ما *n* کار داشته باشیم و رئوس پیمایش شده را، در لحظه بازگشت از آنها، از بزرگتر به کوچکتر، شماره گذاری کنیم
- در واقع نوبت اجرای کار های را با رعایت پیشنیاز های آن، تعیین کرده ایم
- بترتیب بدست آمده برای انجام کارها *Topological Order* گفته میشود که مشخصا پاسخ منحصر به فرد نیست
- برای شروع الگوریتم از یک راس دلخواه شروع میکنیم و الگوریتم *Topological-Sort()* خود را (که علاوه بر عمل *DFS* عمل شماره گذاری را نیز انجام میدهد) بر روی راس انتخابی پیدا میکنیم
- عمل فوق را تا مشاهده (*Visited*) شدن تمامی رئوس تکرار میکنیم
- حل مثال خواسته شده :

(دایره ها نشان دهنده محل شروع عمل *Topological-Sort()* از یک راس دلخواه میباشند و ترتیب اینکار، برای درک بهتر نحوه کار الگوریتم فوق ، از ۱ تا ۵ مشخص شده است)



سوال چهار(*) :

ترتیب نوشتن مدارک روی تابلو ها بصورت زیر است:

تابلوی ۱: G، I، K، S، A، D، R

تابلوی ۲: H، J، L، F، B، E، U

تابلوی ۳: O، C، N

تابلوی ۴: P، M

تابلوی ۵: Q، T

مدارک گفته شده باید بر روی تابلو های گفته شده نوشته شوند و ترتیب اثبات آن ها بصورت زیر است :

G, H -> O

I, J -> P

K, L -> Q

O, P, Q -> S

S, F -> T

A, B, C -> M

D, E -> N

M, N -> R

R, T -> U