

بسمه تعالیٰ



# پروژه پایانی درس ساختمان داده

استاد درس:

مهندس الهام افشار

طراحان:

TA گروه

بهار 1402 - دانشگاه بولنی سینا

# فهرست



۳	مقدمه
۱۴	شرح پروژه
۶	بخش های مختلف پروژه
۶	تعریف جهان
۶	مقداردهی جهان
۷	ایجاد و تکمیل جداول مسیریابی هر نود
۷	درخواست یافتن دو مسیر
۹	بخش امتیازی پروژه
۱۰	نکات پیاده‌سازی
۱۱	نکات مربوط به پروژه
۱۲	اندکی درباره cypher



## مقدمه

در ادامه تمرین اختیاری پیدا کردن کوتاهترین مسیر هم!، از آن زمان تا کنون پیچیدگی‌های جدیدی به مسیرهای کهکشانی اضافه شده است که از جمله این پیچیدگی‌ها نسبی بودن زمان در هر کهکشان و یافتن کوتاهترین مسیر درون‌کهکشانی و بین‌کهکشانی است!

هدف از این پروژه، کار با ساختمان‌داده‌هایی مانند صف و پشته، و کار با گراف و یافتن کوتاهترین مسیر ممکن در گراف و زبان `cypher` است!

در این پروژه، با تعریف نودها و یال‌های هر کهکشان، باید هزینه‌ی کوتاهترین مسیر بین دو نod را طبق شرایط ذکر شده برای ارسال دیتا یافته و چاپ کنید!



# شرح پروژه

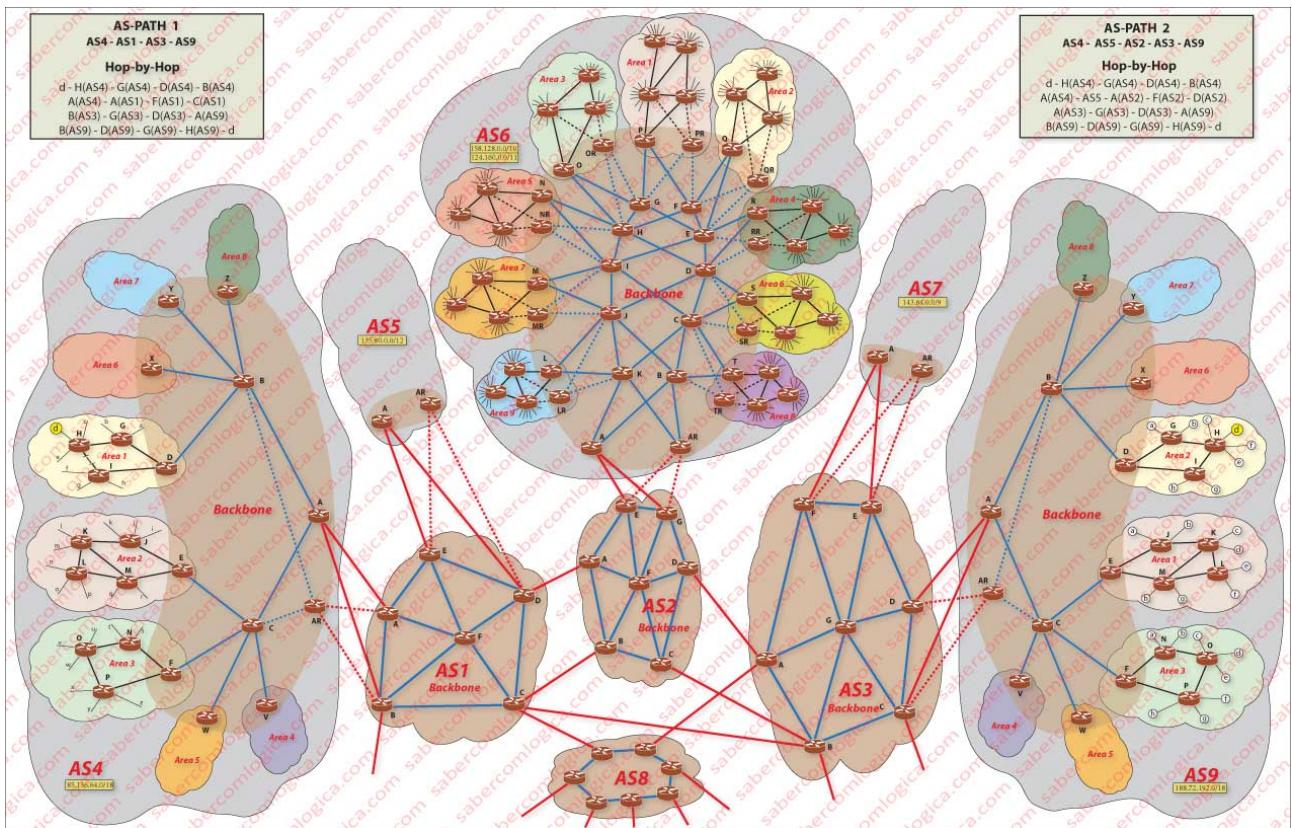
در سیستم مسیریابی جدید هم، مفهومی به نام نود لبه (border gateway node) و نود غیر لبه (non border gateway node) وجود دارد که در ادامه ویژگی‌های هر یک از این نودها را می‌بینید!

همچنین در جهان متصور، شما می‌توانید کهکشان‌هایی شامل این نودها که در واقع سیارات و ستاره‌های ما فرض شده‌اند را تعریف کنید. (هر کهکشان را نیز یک AS می‌نامیم)

دسته اول نود‌های لبه یا BG node نودهایی هستند که در لبه آن کهکشان قرار دارند؛ در واقع این نودها دروازه‌ای از یک کهکشان به کهکشان دیگر هستند و صرفاً از طریق این نودهای لبه می‌توان به آن کهکشان سفر کرد. این نودها الگوریتم‌های مسیریابی درون کهکشانی و بین کهکشانی را اجرا خواهند کرد.

دسته دوم نود غیر لبه یا non BG node نودهایی هستند که در لبه کهکشان قرار ندارند. در واقع هیچ مسیر مستقیمی از این نودها به نودهای کهکشان دیگر وجود ندارد. این نودها صرفاً الگوریتم‌های مسیریابی درون کهکشانی را اجرا خواهند کرد.





برای مشاهده تصویر با کیفیت [اینجا](#) کلیک کنید.

برای درک بهتر از جهان، تصویر بالا را مشاهده کنید که شامل ۹ کهکشان با نام‌های فرضی AS1 - AS9 را نمایش می‌دهد که در هر کهکشان تعدادی نود وجود دارد.

طبق تعریف نودهای AS3، همگی نود لبه (BG NODE) هستند.  
نود G از همین کهکشان یک نود غیر لبه است چرا که به صورت مستقیم به هیچ نودی از کهکشان دیگر وصل نشده است!

- توجه شود که همه یال‌های ارتباطی باید دارای وزن مثبت غیر صفر باشند و اعتبار سنجی ورودی کاربر بر عهده شما است.

# بخش های مختلف پروژه

## ۱. تعریف جهان

با استفاده از دستورات زبان CYPHER، جهانی را شامل موارد زیر ایجاد کنید:

- جهان شما باید حداقل 3 کهکشان مرتبط شده<sup>۱</sup> داشته باشد، که هر کهکشان حداقل 5 نود دارد.

```
CREATE (A: Node, {id: '1', Type: 'BG'})  
CREATE (B: Node, {id: '2', Type: 'NON-BG'})
```

برای مثال این 2 دستور، دو نود A و B را ایجاد میکند که در هر نود شناسه منحصر به فرد و تایپ دارد.

توجه : نامگذاری و دارا بودن صفات شناسه برای هر نود الزامی است و سایر موارد اختیاری است.  
توجه: نود ایزوله در یک کهکشان و کهکشان ایزوله در جهان نداریم!

## ۲. مقداردهی جهان

هزینه تمام یال‌های موجود در جهان را مشخص کنید.  
همانطور که گفته شد، هزینه هر یال یک عدد مثبت غیر صفر است.  
اگر لینکی حذف شد می‌توانید از هزینه بینهایت یا nan و ... برای نمایش آن به دلخواه استفاده کنید.

```
CREATE (A) - [:ROAD {cost:50}] -> (B)
```

دستور بالا هزینه بین نود A به B را برابر با 50 قرار می‌دهد. ( دقیت شود که گراف جهت دار نیست و بین هر دو نود متمایز تنها یک هزینه متصور است )

---

<sup>۱</sup>منظور از مرتبط بودن کهکشان‌ها این است که هیچ کهکشانی نباید ایزوله باشد ( گراف جهان شما باید همبند باشد )

## ۳. ایجاد و تکمیل جداول مسیریابی هر نود

در هر نود لبه (BG) در کهکشان، باید حداقل دو جدول داشته باشد که جدول اول شامل کوتاهترین مسیر از آن نود با تمامی نودهای درون آن کهکشان؛ و جدول دوم شامل مسیرهای مستقیم از آن نود به نودهای لبه در کهکشان دیگر است. به بیان دیگر هزینه‌ی نود لبه تا نود لبه کهکشان دیگر مشخص شود.

همچنین هر نود غیر لبه (NON BG) یک جدول مسیریابی بیشتر ندارد و آن هم شامل کوتاهترین مسیر از آن نود به تمامی نودهای درون آن کهکشان است.

### نکته

- اجرای الگوریتم‌های تکمیل جداول مسیریابی هر نود تنها یکبار بعد از مقداردهی جهان شما است! و با دستور update costs انجام می‌شود.
- پس از ویرایش هزینه یک یال، تمامی این الگوریتم‌ها باید دوباره اجرا شوند تا جداول آپدیت شوند و کوتاهترین مسیر صحیح پیدا شود.
- بدیهی است که جداول نودها باید همگرا باشند یعنی نودها باید درک یکسانی از کوتاهترین مسیر داشته باشند.

## ۴. درخواست یافتن دو مسیر

توجه: هزینه مجموع وزن یال‌های موجود در آن مسیر که باید به ترتیب در خروجی چاپ شود.

### الف) یافتن هزینه کوتاهترین مسیر درون کهکشانی بین دو نود

دستور مد نظر:

FIND AS4 .W->AS4 .X

خروجی مد نظر:

W->C->A->B->X : TOTAL\_COST

بعد از چاپ کوتاهترین مسیر ممکن بین آن دو نود، مجموعه هزینه را نیز چاپ کنید.

## ب) یافتن هزینه کوتاه‌ترین مسیر بین کهکشانی بین دو نود

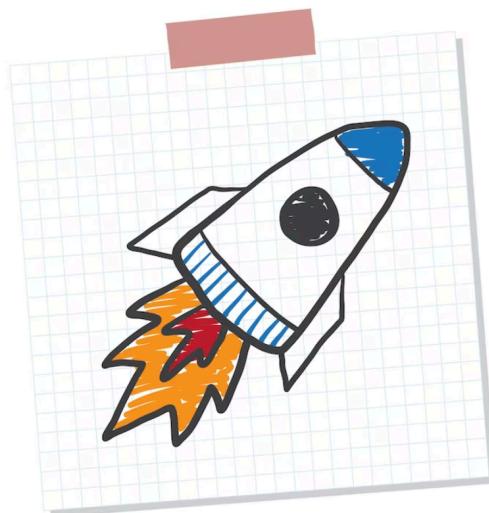
دستور مد نظر:

**FIND AS4.W->AS1.F**

خروجی مد نظر:

**W->C->AR->A->F : TOTAL\_COST**

دستور مدنظر درخواست یافتن کوتاه‌ترین مسیر بین نود W از کهکشان AS4 و نود F از کهکشان AS1 را مطابق تصویر دارد. بنابراین نود W AS4.N زدیک‌ترین نود لبه‌ای را که به کهکشان AS1 متصل باشد را یافته (که در اینجا نود AS4.AR است و طبق نکته سیب‌زمینی داغ<sup>2</sup>، ارسال از طریق نود AS4.A پرهزینه‌تر فرض شده است). سپس از طریق این لینک دیتا را به نود A AS1.A و سپس به نود F AS1.F که مقصد است می‌فرستد.



---

<sup>2</sup> توضیحات مربوط به این نکته را در قسمت توضیحات در آخر سند مطالعه فرمایید.

## بخش امتیازی پروژه

- تمیز بودن کد و گزارش کار و کامنت‌گذاری صحیح و مناسب کد (5 امتیاز)
- پیاده سازی بیشتر (۲۰ امتیاز)

در یک سیاست کلی، جداول مسیریابی هر نود از طریق تبادل پیام با نودهای همسایه که مستقیماً به آنها وصل است تکمیل می‌شود. یعنی هر نود، کل جدول مسیریابی خودش را برای همسایه‌اش می‌فرستد!<sup>۳</sup>

در حالت دیگر، یک دید کلی نسبت به جهان وجود دارد؛ یعنی اجرای الگوریتم مسیریابی با توجه به توپولوژی هر کهکشان انجام می‌شود.<sup>۴</sup> تفاوت این دو روش در [اینجا](#) آمده است.

• پیاده سازی (semantic graph) گراف معنایی (semantic graph) : به دلخواه یک سیستم را بصورت مجموعه‌ای از موجودیت و روابط بین آنها پیاده سازی کنید ، یعنی نود‌ها موجودیت‌های شما و یال‌ها روابط بین این موجودیت‌ها هستند .

برای مثال دو دستور زیر بخشی از پیاده سازی سازمان یک شرکت نوعی است :

```
CREATE (e1: EMPLOYEE, {Empid: '1', Lname: 'Smith', Fname: 'John'})  
CREATE (d1: DEPARTMENT, {Dno: '5', Dname: 'Research'})  
CREATE (e1) - [ : WorksFor ] -> (d1)
```

مثال بالا برای نمایش موجودیت کارمند که دارای ویژگی‌های کد پرسنلی، نام، و نامخانوادگی است از یک نود با نام e1 استفاده شده است. همچنین دیپارتمان با نام d1 دارای شماره دیپارتمانی 5 است و برای تحقیقات به کار می‌رود.

نهایتاً با ایجاد یال بین این دو نود مشخص می‌شود که کارمند e1 برای بخش d1 کار می‌کند؛ و بخش d1 دارای کارمندی به نام e1 است.

<sup>۳</sup> در شبکه‌های کامپیوتری به این پروتکل DISTANCE VECTOR می‌گویند.

<sup>۴</sup> در شبکه‌های کامپیوتری به این پروتکل LINK STATE می‌گویند.

# نکات پیاده‌سازی

## نکته سیب‌زمینی داغ

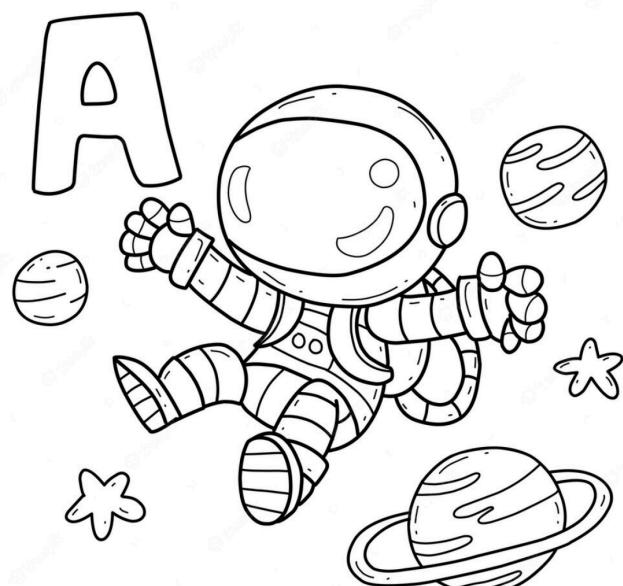
طبق اصل HOT POTATO یا سیب‌زمینی داغ! یک نود غیر لبه برای ایجاد ارتباط با کهکشان دیگر، دیتای خودش را به نزدیک‌ترین نود لبه از کهکشان خودش که متصل به آن کهکشان باشد می‌فرستد و کاری به بهینه‌بودن یا نبودن سایر مسیرها ندارد. این یعنی تنها کوتاه‌بودن مسیر درون کهکشانی برایش اهمیت دارد و توانایی اجرای این الگوریتم را نیز دارد.

### نکته ۱

برای نگه داشتن لیست وزن یال‌های میسر، حتما از ساختمان‌داده صف یا پشته استفاده کنید.

### نکته ۲

با استفاده از اعتبارسنجی ورودی، تضمین کنید که نام نودهای یک کهکشان یکتا هستند و از درج نود تکراری جلوگیری شود.  
وجود نودهای همنام در کهکشان‌های مختلف مجاز است؛ و با عملگر نقطه مشخص می‌شود برای مثال: AS.W



# نکات مربوط به پروژه

پروژه دارای **45 کات آف** درصدی است.

بارم کلی پروژه 100 امتیاز به همراه ۲۵ امتیاز اضافه برای بخش امتیازی است.(از دو بخش اختیاری یکی را به دلخواه انجام دهید )

نوشتن گزارش کار و ارائه آنلاین الزامی است. •

• تنها زبان های C و C++ برای انجام پروژه مجاز بوده و استفاده از گیت الزامی است.

در تیم های دونفر کامیت های هر شخص جداگانه مورد بررسی قرار خواهد گرفت.  
کامیت ها باید با کامنت های صحیح و معنادار ارسال شوند.

• استفاده از برنامه نویسی شی گرا OOP لازم است یعنی برنامه شما باید حداقل کلاس های جهان (universe) و کهکشان (galaxy) و نود (node) را دارا باشد .

• حتما فایل اجرایی پروژه خود را (exe , ... ) را به همراه گزارشکار پروژه ارسال کنید  
پروژه شما با تست کیس زیر ارزیابی می شود :

ایجاد ۳ کهکشان دارای ۵ نود و وزن دهی به یال ها و در نهایت مسیر یابی بین نود ها

گروهها حداقل می توانند 2 نفره باشند. نام و شماره دانشجویی افراد گروه خود را به ایمیل **HM2100MH@gmail.com** ارسال فرمایید. ارسال ایمیل توسط یکی از اعضای گروه کافی می باشد.

**مهلت ارسال پروژه تا 28 مرداد می باشد.**



# اندکی درباره cypher

در پایگاهدادههای مبتنی بر گراف (graph based database)، برای مدلسازی موجودیت‌ها (entity)، و روابط بین‌شان (relationships) می‌توان از مجموعه‌ای از نودها و یال‌ها استفاده کرد. در واقع نودها، موجودیت‌ها هستند؛ و یال‌ها روابط بین آنها. حال زبان cypher که یک زبان برای ایجاد کوئری‌های توصیفی (high-level declarative) است، تنها با مشخص کردن خواسته‌اش، گراف‌های متنوعی را ایجاد می‌کند.

برای آشنایی بیشتر با پایگاهدادههای گرافی این [لینک](#) را مطالعه فرمایید.

موفق و پیروز باشید (:)