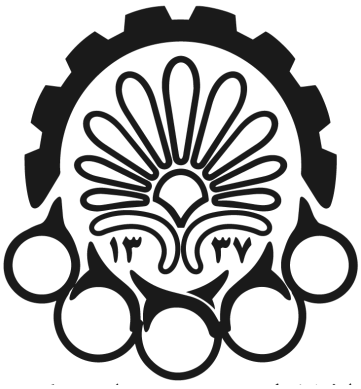


تمرین امتیازی سوم درس آزمون نرم افزار

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)

بردیا اردکانیان

۹۸۳۱۰۷۲

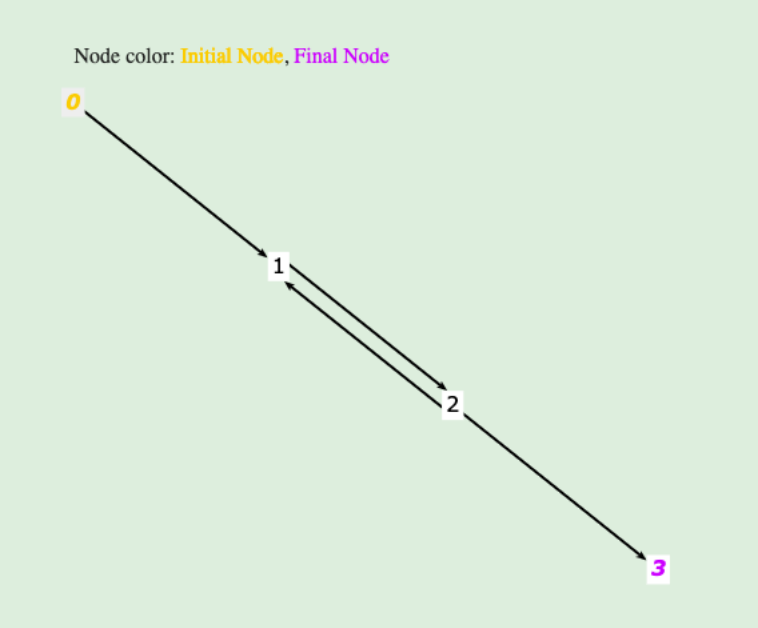
استاد درس: دکتر گوهری

بهار ۱۴۰۲

7.2.2-q8: Design and implement a program that will compute all prime paths in a graph, then derive test paths to tour the prime paths. Although the user interface can be arbitrarily complicated, the simplest version will be to accept a graph as input by reading a list of nodes, initial nodes, final nodes, and edges.

در این برنامه ابتدا یک گراف ورودی گرفته می‌شود. گراف در فرمت لیست همسایگی پذیرفته می‌شود. در مرحله اول با توجه به گره‌هایی که در گراف وجود دارد تمام ترکیب‌های (مبدا، مقصد) دودویی تشکیل می‌شوند و تابع findPrimePaths برای هرکدام از این جفت‌ها صدا زده می‌شود. علت جفت بودن پیدا کردن حلقه در برنامه می‌باشد.

مثال زیر را در نظر بگیرید:



در این گراف مسیرهای prime زیر وجود دارند:

$$[1, 2, 1], [2, 1, 2], [0, 1, 2, 3]$$

الگوریتم پیدا کردن مسیرهای prime از یک dfs ساده استفاده می‌کند و قادر به پیدا کردن تمامی حلقه‌ها با یک بار صدا زده شدن نیست و باید backtracking انجام دهد. با این حال برای پیدا کردن حلقه‌ها کماکان باید مبدا مقصد ۱ و ۲ به عنوان ورودی داده شود وگرنه حلقه را در مسیر $[0, 1, 2, 1]$ پیدا می‌کند که غلط می‌باشد. پس باید جفت ترکیب‌های مبدا مقصد دودویی ایجاد شود تا ترکیبی را از قلم نیاندازد.

منتهی این اشکال جدیدی را به وجود می‌اورد و آن مسیرهایی می‌باشد که در مسیرهای دیگر وجود دارند. به عنوان مثال با مبدا مقصد ۰ و ۱ مسیر $[0, 1]$ را داریم که زیر مجموعه مسیر $[0, 1, 2, 3]$ می‌باشد. با کمک توابع نوشته شده مسیرهایی که در مسیرهای کامل وجود دارند حذف می‌شوند. در انتهای این مرحله تمام مسیرهای prime تشکیل شده‌اندو

در مرحله آخر باید با کمک مسیرهای prime، مسیر تست ایجاد کنیم تا مسیرهای prime را طی بکند. برای این کار ابتدا مسیرهای کامل را پیدا می‌کنیم (مسیرهایی که از مبدا به مقصد می‌رسند). تمامی مسیرهای کامل مسیرهای تست نیز می‌توانند باشند (مگر اینکه مسیر تست بزرگتری پیدا شود که علاوه بر مسیر کامل مسیرهای حلقوی را نیز طی کند). بعد از پیدا کردن مسیرهای کامل، مسیرهای حلقوی را پیدا می‌کنیم.

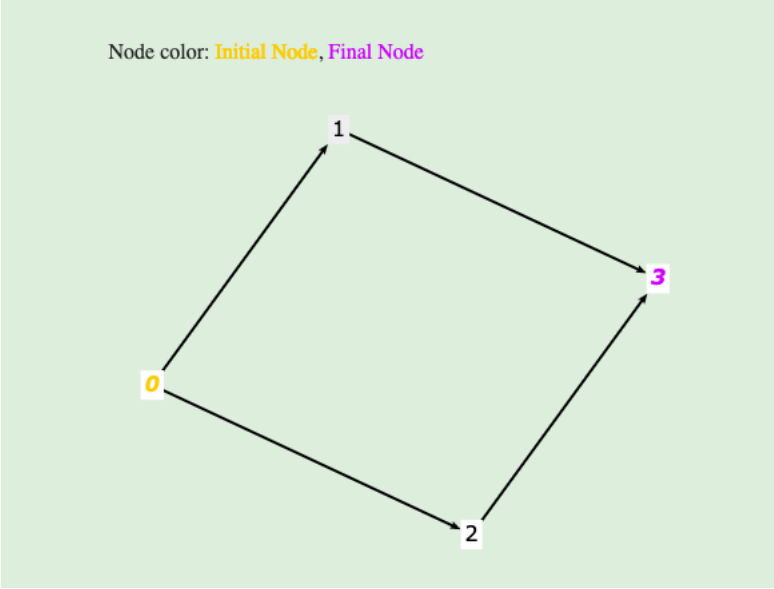
در مرحله بعدی مسیرهای حلقوی را در مسیرهای کامل اضافه می‌کنیم. به عنوان مثال یک مسیر کامل $[0, 1, 2, 3]$ و یک مسیر حلقوی $[1, 2, 1]$ داریم. با اضافه کردن مسیر حلقوی در شمارنده ۱ مسیر کامل، مسیر جدیدی خواهیم داشت که علاوه بر کامل بودن، یک یا چند مسیر حلقوی دیگر را پوشش می‌دهند. در مثال قبل خروجی مسیر $[0, 1, 2, 1, 2, 3]$ می‌شود که هر دو مسیرهای حلقوی موجود را پوشش می‌دهد. علاوه بر این مسیر، مسیر کامل اولیه $[0, 1, 2, 3]$ نیز در لیست مسیرهای تست وجود دارد که پاسخ هم همین می‌باشد.

خروجی برنامه:

```
Prime Paths: [[2, 1, 2], [0, 1, 2, 3], [1, 2, 1]]
Test Paths: [[0, 1, 2, 1, 2, 3], [0, 1, 2, 3]]

Process finished with exit code 0
```

برای تست بیشتر برای گراف زیر نیز الگوریتم را تکرار می‌کنیم.



مسیرهای تست و مسیرهای prime به شرح ذیل می‌باشد:

2 test paths are needed for Prime Path Coverage using the prefix graph algorithm

Test Paths	Test Requirements that are toured by test paths directly
[0,1,3]	[0,1,3]
[0,2,3]	[0,2,3]

خروجی برنامه:

```
Prime Paths: [[0, 2, 3], [0, 1, 3]]
Test Paths: [[0, 2, 3], [0, 1, 3]]

Process finished with exit code 0
```

که نشان می‌دهد الگوریتم نوشته شده به درسی کار می‌کند. الگوریتم پیچیدگی محاسباتی $O(MN)$ دارد که M تعداد گره‌ها و N تعداد یال‌ها می‌باشد.