دانشکده مهندسی کامپیوتر

جزوه درس
ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۲۹

Topological Sort - SCCs

سید مصطفی مسعودی - یاسین عسکریان - ۱۳۹۸/۱۰/۰۷

۱.۲۹ مرتب سازی موضعی (Topological Sort)

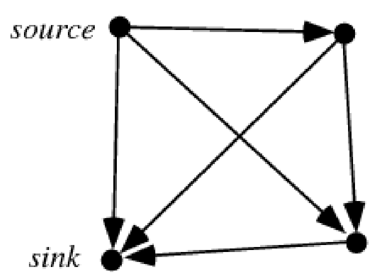
مرتب سازی توپولوژیک، مرتب سازی رئوس یک گراف جهت دار بدون طوقه [1] و بدون دور (DAG) است به طوری که هر راس قبل از رئوسی میاید که به آنها یال خروجی داده است.
کاربرد اصلی مرتب‌سازی موضعی در زمانبندی یک سلسله‌ای از کارها یا وظایف است. برای مثال در زمان شستن لباس‌ها، قبل از شروع خشک شدن لباس‌ها، کار شستن باید تمام شود

۱.۱.۲۹ Source

گره ای است که ، هیچ یالی به آن وارد نشده است .

۲.۱.۲۹ Sink

گره ای است که ، هیچ یالی از آن خارج نشده است .



شکل ۱.۲۹: Source and Sink

[1] طوقه : گره ای که یالی از خود از خودش به خودش وجود داشته باشد

۳.۱.۲۹ الگوریتم پایه

- بر روی گراف DFS زده تا گره Sink پیدا شود
- آن را به انتهای لیست اضافه کرده
- آن را از گراف حذف کنید
- دوباره این مراحل را تا تمام شدن گره ها تکرار کنید

```
LinearOrder(G)

while G non-empty:
  Follow a path until cannot extend
  Find sink v
  Put v at end of order
  Remove v from G
```

شکل ۲.۲۹: الگوریتم پایه

اردر الگوریتم پایه

- هر بار اجرای الگوریتم DFS با اردر تعداد گره ها طول میکشد O(V)
- و برای هر گره ،مرحله ی قبل را باید اجرا کرد،.یعنی به تعداد V بار تکرار می کنیم
- پس اردر کلی الگوریتم O(V*V) می شود

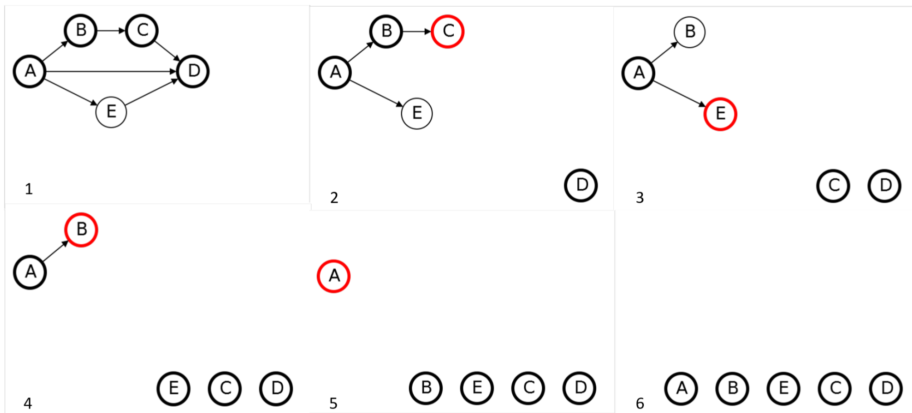
۴.۱.۲۹ الگوریتم سریعتر

در این حالت ، تنها یک بار DFS را اجرا کنید و بر اساس شماره ی PostVisit به ترتیب از کوچک به بزرگ به انتهای لیست اضافه کنید. در این حالت فقط یک بار الگوریتم پیمایش عمق اول اجرا خواهد شد و اردر برنامه بهتر خواهد شد.

```
TopologicalSort(G)

DFS(G)
sort vertices by reverse post-order
```

شکل ۳.۲۹: الگوریتم سریع تر

مراحل اجرای الگوریتم مرتب سازی موضعی،به صورت زیر است :



شکل ۴.۲۹: مرتب سازی موضعی

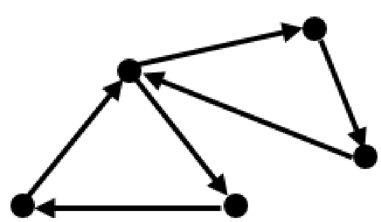
۲.۲۹ گراف قویا همبند

۱.۲.۲۹ جفت راس قویا همبند

در گراف جهت‌دار دو راس u و v قویا همبند هستند، اگر مسیری از u به v و مسیری از v به u وجود داشته باشد.

۲.۲.۲۹ گراف قویا همبند

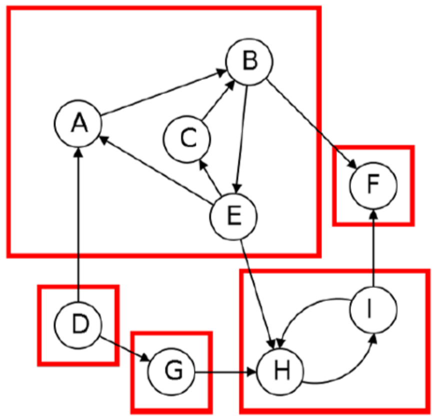
یک گراف جهت دار قویا همبند است اگر هر دو راس آن قویا همبند باشند. در تصویر می‌توانید یک گراف قویا همبند را مشاهده کنید.



شکل ۵.۲۹: گراف قویا همبند

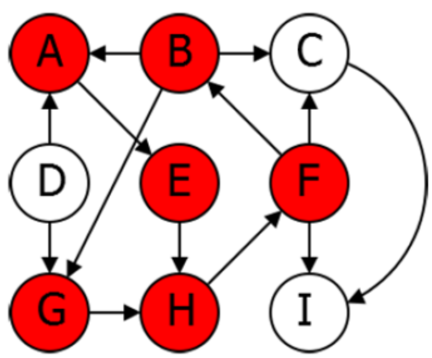
۳.۲.۲۹ مؤلفه‌ی قویا همبند [۲]

اگر یک زیر مجموعه‌ای از رئوس گراف جهت‌دار G همراه با تمام یال‌های بین آنها(یک زیرگراف) که خاصیت قویا همبندی بین هر دو راس آن وجود دارد، قابل گسترش نباشد، یک مؤلفه‌ی قویا همبند در گراف جهت‌دار G است. قابل گسترش نبودن به این معنی که نتوان هیچ راسی به این زیرمجموعه اضافه کرد که همچنان این زیرمجموعه خاصیت قویا همبندی خود را حفظ کند. رئوس هر گراف جهت‌داری را می‌توان به تعدادی مؤلفه‌ی قویا همبند افراز کرد.



شکل ۶.۲۹: افراز یک گراف به زیرگراف های قویا همبند

در شکل زیر،گره هایی که با گره A در یک زیرگراف قویا همبند هستند ،با رنگ قرمز نشان داده شده است.



شکل ۷.۲۹: یک زیرگراف قویا همبند

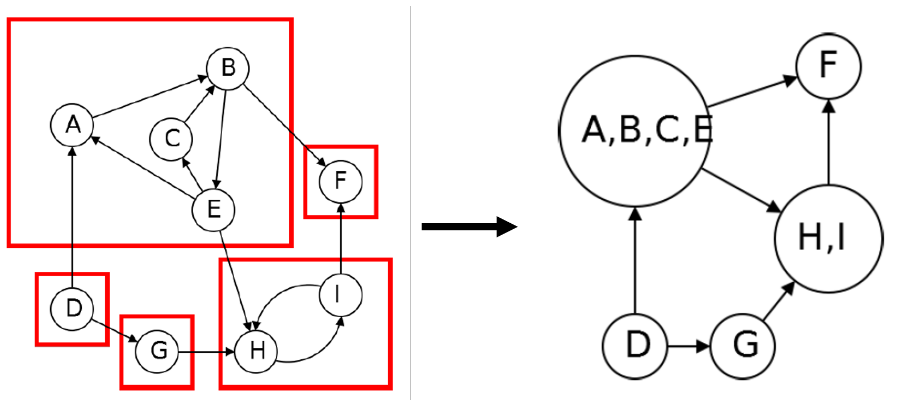
[۲] Strongly Connected Component (SCC)

۴.۲.۲۹ گراف معکوس

گراف معکوس، گرافی است که برای گراف‌های جهت‌دار تعریف می‌شود.به عبارت ساده‌تر گراف معکوس G ،همان گراف G است که جهت یال‌هایش عکس شده.
ویژگی قویا همبندی بین هر دو جفت راس بعد از معکوس کردن حفظ می‌شود. پس ویژگی قویاهمبندی گراف در گراف معکوس نیز حفظ می‌شود.
به صورت کلی‌تر مولفه‌های قویا همبند گراف و گراف معکوس یکیست.

۵.۲.۲۹ Metagraph

برای ساختن Metagraph ،باید مؤلفه های قویا همبند یک را پیدا کرد و روابط بین آن ها را درنظر گرفت.
به شکل زیر دقت کنید.



شکل ۸.۲۹: Metagraph

و می توان گفت،همیشه Metagraph مربوط به یک گراف ، یک گراف جهت دار بدون دور (DAG) است.

۶.۲.۲۹ الگوریتم ساده

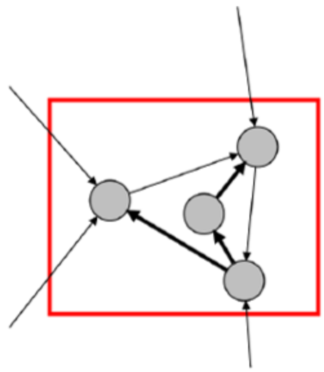
```
EasySCC(G)

for each vertex v:
  run explore(v) to determine
    vertices reachable from v
for each vertex v:
  find the u reachable from v that
    can also reach v
these are the SCCs
```

Runtime $O(|V|^2 + |V||E|)$.

شکل ۹.۲۹: الگوریتم ساده برای پیدا کردن مؤلفه های قویا همبند در یک گراف

به دلیل بالا بودن پیچیدگی زمانی این الگوریتم به یک الگوریتم سریعتر نیاز خواهیم داشت.

Sink Component



شکل ۱۰.۲۹: Sink Component

برای پیدا کردن مؤلفه های قویا همبند در گراف، می توان هر بار مؤلفه ی Sink را پیدا کرد، و آن را از گراف جدا کرده و مجددا به دنبال مؤلفه ی Sink بعدی گشته و این کار را تکرار می کنیم تا تمام مؤلفه ها بدست آیند.

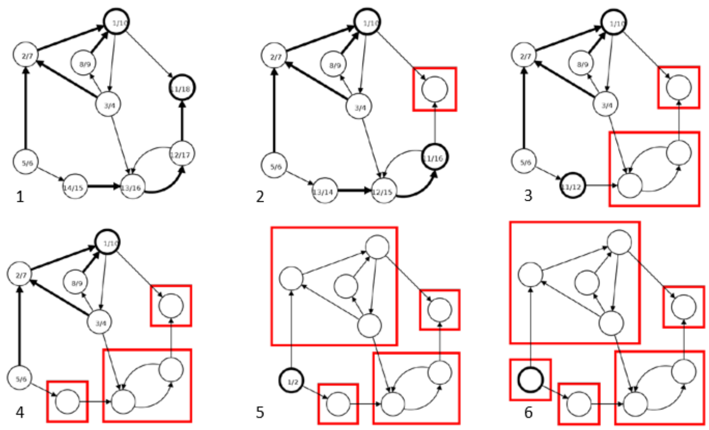
۷.۲.۲۹ الگوریتم پایه

الگوریتم DFS را روی گراف اجرا کرده و PreVisit و PostVisit را برای گره ها می نویسیم.آن گره ای که بزرگترین شماره یPostvisit را دارد، قطعا در مؤلفه ی Source قرار دارد. پس برای بدست آوردن مؤلفه های Sink ، باید مؤلفه های Source در گراف معکوس را بدست آورد.

```
SCCs(G)
run DFS(G^R)
let v have largest post number
run Explore(v)
vertices found are first SCC
Remove from G and repeat
```

شکل ۱۱.۲۹: الگوریتم پایه

می توانید پیمایش یک گراف برای پیدا کردن مؤلفه های قویا همبند از طریق الگوریتم ساده را در شکل زیر مشاهده کنید



شکل ۱۲.۲۹: پیمایش گراف از طریق الگوریتم پایه

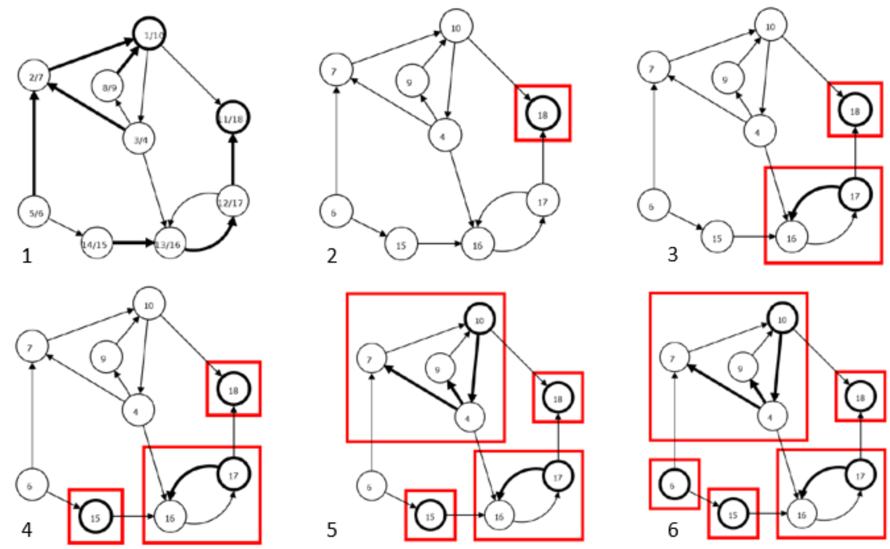
۸.۲.۲۹ الگوریتم سریع

همانطور که مشاهده کردید در الگوریتم پایه پس از هر بار اجرای DFS و پیدا کردن مؤلفه های همبند،گره های آن مؤلفه حذف شده و دوباره برای پیدا کردن بزرگترین شماره ی PostVisit گراف جدید تمام آن مراحل از اول اجرا شده است اما در الگریتم سریع تنها با یک بار اجرای DFS و یادداشت کردن کردن شماره ی PostVisit های گراف معکوس،جستجو را برای پیدا کردن مؤلفه های قویا همبند از بزرگ ترین شماره ی PostVisit، از گراف معکوس،در گراف اصلی شروع کرده و پس از پیدا کردن هر مؤلفه قویا همبند گره های مشاهده شده را یادداشت کرده و دوباره از بزرگ ترین شماره ی PostVisit مشاهده نشده جستجو را آغاز کنید

```
SCCs(G)

Run DFS(G^R)
for v ∈ V in reverse postorder:
  if not visited(v):
    Explore(v)
    mark visited vertices
      as new SCC
```

شکل ۱۳.۲۹: الگوریتم سریع

می توانید پیمایش یک گراف برای پیدا کردن مؤلفه های قویا همبند از طریق الگوریتم سریع را در شکل زیر مشاهده کنید



شکل ۱۴.۲۹: پیمایش گراف از طریق الگوریتم سریع

زمان اجرای الگوریتم سریع $O(|V|+|E|)$ می باشد
[۱]

Bibliography

[1] https://opedia.ir/%D8%A2%D9%85%D9%88%D8%B2%D8%B4/%D8%A7%D9%84%DA%AF%D9%88%D8%B1%DB%8C%D8%AA%D9%85/%D9%85%D8%B1%D8%AA%D8%A8_%D8%B3%D8%A7%D8%B2%DB%8C_%D8%AA%D9%88%D9%BE%D9%88%D9%84%D9%88%DA%98%DB%8C%DA%A9.