دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۲۷

پیمایش در گراف

ملیکا احمدی رنجبر و رضا علیدوست - ۱۳۹۸/۹/۳۰

۱.۲۷ نمایش گراف و انواع گراف

هرگراف شامل دو عنصر گره (node) و یال (edge) است . سه نوع نمایش گراف موجود است :

- لیست یال list of edge شامل تمام یال های بین دو گره
- ماتریس مجاورت adjacency matrix ماتریسی از ۰ و ۱ که اگر یال بین دو گره باشد ۱ و اگر نه ۰ میگذاریم
- لیست مجاورت adjacency list لیست گره های متصل به هر گره عملیات های متفاوت سرعت های متفاوتی در هر نوع نمایش دارند و این بستگی به نوع گراف دارد. دو نوع گراف موجود است :
- dense graph تعداد یال های زیادی دارد |E| هم ارز $|V|^2$

  sparse graph تعداد یال های کمی دارد |E| هم ارز |V|

۲.۲۷ پیمایش گراف

پیمایش گراف در حالت کلی به معنی گذشتن و دیدن تمام راس‌های گراف است و معمولا پیمایش از طریق یال‌های موجود در گراف انجام می‌شود. معمولا پیمایش گراف به تنهایی ارزش خاصی ندارد و هدف از پیمایش، محاسبه یا پیدا کردن چیز خاصی است.

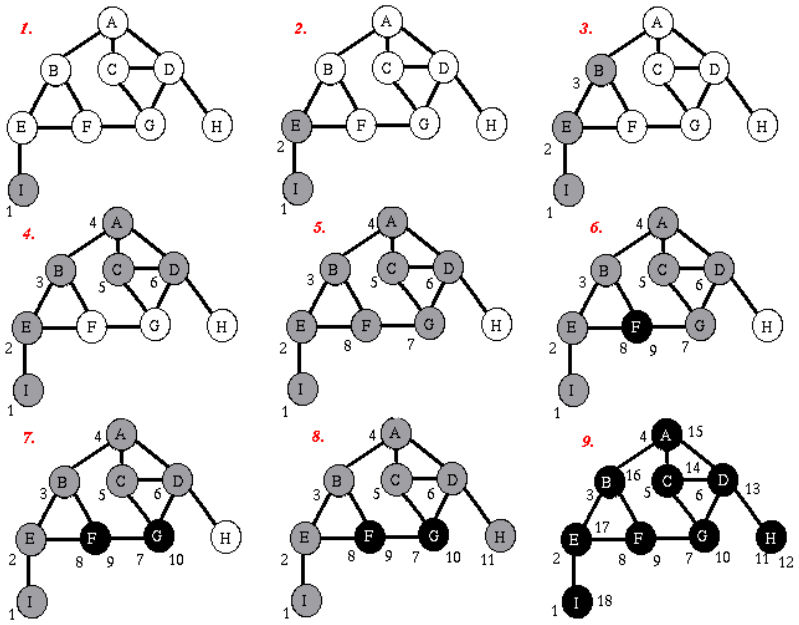
Explore ۱.۲.۲۷

با Explore کردن گره v در گراف G همه گره‌هایی که گره v به آن دسترسی دارند را پیدا میکنیم. به عبارتی همه‌ی گره‌هایی که با گره v دریک مولفه همبندی هستند را بدست می‌آوریم. الگوریتم به این شکل است که ابتدا یک راس مانندv انتخاب میکنیم و آن را ریشه می‌نامیم. ریشه را علامت‌گذاری می‌کنیم. سپس یک راس دلخواه علامت نخورده مجاور با v را انتخاب می‌کنیم و آن را u می‌نامیم. u را یکی از بچه‌های v می‌کنیم،

سپس u را علامت می‌زنیم. حال همین الگوریتم را روی u از ابتدا اجرا میکنیم. الگوریتم گفته شده زمانی به بن‌بست می‌خورد که به ازای راسی مانند u ، تمام همسایه‌هایش علامت خورده باشند. در این صورت به راس پدر برمی‌گردیم و دوباره همین الگوریتم را از ادامه اجرا می‌کنیم. برنامه زمانی متوقف می‌شود که به ریشه برگشته باشیم و تمام همسایه هایش علامت خورده باشند که در این صورت می‌گوییم الگوریتم پایان یافته است. شبه‌کد ۱ آن به صورت زیر است :

Result: Explore(V)
visited(V) $\leftarrow$ *true*
for *(V,W)* $\in$ *E* ***do***
  if not visited(W) ***then***
    Explore(W);
  end
end

Algorithm 1: *Explore(V)*



شکل ۱.۲۷: Explore(V)

۲.۲.۲۷ جست‌وجوی اول عمق (DFS)

جست‌وجوی عمق اول که به DFS(DepthFirstSearch) معروف است در واقع الگوریتمی برای پیمایش کل گراف است. اگر گراف همبند نباشد Explore(V) تنها راس‌های مولفه همبندی ریشه را پیمایش می‌کند پس برای پیمایش روی تمام راس‌ها باید Explore به ازای هر راس علامت نخورده تکرار شود.شبه‌کد ۲ آن به

صورت زیر است:

Result: DFS(V)
for *all* $v \in V$ ***do***
| $visited[v] \leftarrow false$
end
for v $\in V$ **do**
| **if** *not visited[v]* **then**
| | Explore(v)
| **end**
end

Algorithm 2: DFS(V)

جست‌جوی اول عمق یال‌هایی که تشکیل دور می‌دهند را نمی‌رود در نتیجه اگر یال‌های رفته شده را کنار هم بگذاریم، تشکیل یک درخت ریشه دار می‌دهند که به آن درخت DFS می‌گویند.

۳.۲.۲۷ Previsit and Postvisit Orders

زمان ورود و خروج راس نیز ویژگی‌های منحصر به فردی دارد که این‌گونه تفریف میگردند :

- Previsit Order : زمانی که برای اولین بار وارد یک راس می‌شویم و آن را علامت‌گذاری میکنیم.
- Postvisit Order : زمانی که برای آخرین بار راس را می‌بینیم و از آن خارج می‌شویم و تمام همسایه‌هایش دیده شده است و در حال بازگشت به راس پدر هستیم.

اگر در شکل ۲۷.۱ بخواهیم با رنگ‌ها این دو زمان را معادل کنیم، زمان خاکسنری شدن برابر زمان ورود و زمان سیاه شدن برابر زمان خروج است. خاصیت خوبی که این تعاریف می‌دهد این است که ما می‌توانیم فرض کنیم هنگامی که از یک راس خارج می‌شویم، کار تمام زیردرخت آن تمام شده است.

۳.۲۷ مولفه های همبند

می خواهیم ببینیم کدام گره ها در گراف G از بقیه گره ها قابل دسترسی است.
زمانی دو گره از یکدیگر قابل دسترسی اند اگر و تنها اگر هر دو در یک مولفه همبند باشند. قابل دسترسی بودن یک رابطه هم ارزی است :

هر گره به خودش دسترسی دارد

اگر گره u به گرهv قابل دسترسی باشد ، پس گره v هم به گره u دسترسی دارد.

```
Result: DFS(G)
for (v) ∈ Vmarkvunvisited do
    cc ← 1
    for (v) ∈ V do
        if not visited(v) then
            Explore(W)
            cc ← cc + 1;
        end
    end
end
```

۴.۲۷ گراف جهتدار

گراف جهت دار در این گراف هر یال جهت دارد و دارای راس شروع و پایان می باشد. پیمایش DFS در گراف جهت دار : برای این کار فقط اجازه حرکت در جهت یال ها را داریم ، سپس با صدا زدن تابع Explore(v) ، تمام راس های قابل دسترس از راس v را می یابیم .

چرخه در گراف جهت دار اگر از یکی از راس های گراف شروع کنیم و به ترتیب یال ها را بپیماییم به طوریکه راس شروع یال بعدی راس پایانی یال فعلی باشد و به همان راس ابتدایی برسیم میگوییم که در گراف چرخه وجود دارد و هیچ ترتیب خطی از آن وجود ندارد.

dag (directed acyclic graph) : به گراف جهت داری میگویند که چرخه در آن وجود ندارد و میتوان ترتیب خطی از آن به دست آورد.

همچنین سعی کنید حتی‌الامکان به منابع و مراجع مناسب ارجاع دهید [۱]. علاوه بر مراجع چنانچه ابزار یا وب‌سایت قابل توجهی موجود است خوب است به آن هم ارجاع دهید [۲].

Bibliography

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[2] "Opedia.ir." `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%86%D9%85%D8%A7%DB%8C%D8%B4_%DA%AF%D8%B1%D8%A7%D9%81_%D9%87%D8%A7`. Accessed: 2019-12-10.