یادآوری جلسه بیست‌وسوم — جستجوی سطح نخست — عرفان صدرائیه

در جلسه قبل درباره الگوریتم جستجوی سطح نخست(BFS) بحث کردیم و دو کاربرد آن را مورد بررسی قرار دادیم. همچنین مقدمه‌ای از گراف‌های جهت‌دار گفته شد. ابتدا جستجوی سطح نخست را مورد بررسی می‌کنیم:

- **جستجوی سطح نخست:** در این الگوریتم، ابتدا راس i را مشاهده می‌کنیم. سپس همسایه‌های i ، سپس همسایه‌های همسایه‌های آن و این‌کار را تا جایی که تمام راس‌ها مشاهده شوند، ادامه می‌دهیم. پیاده‌سازی این الگوریتم با استفاده از داده ساختار صف به صورت زیر انجام می‌شود:

  ۱. ابتدا راس i وارد صف می‌کنیم.

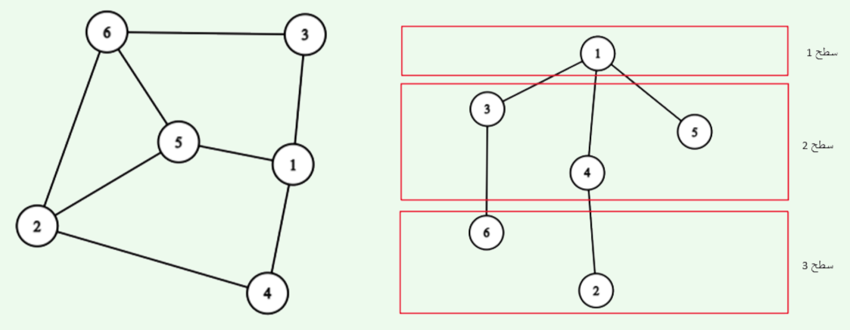
  ۲. راس جلوی صف را مشاهده می‌کنیم و از صف خارج می‌کنیم.

  ۳. همسایه‌های آن راس که هنوز مشاهده نشده‌اند را به انتهای صف اضافه می‌کنیم.

مرحله ۲ و ۳ را تا زمانی‌که تمام راس‌ها مشاهده شوند، ادامه می‌دهیم. شبه کد مربوط به الگوریتم جستجوی سطح نخست به صورت زیر است:

```
BFS(i):
    mark[i]=True
    Q.enque(i)
    dist[i]=0
    father[i]=-1
    while(Q is not empty):
        j = Q.deque()
        //visit j
        for k in N(j):        //j همسایه های راس=N(j)
            if not mark[k]:
                mark[k]=True
                Q.enque(k)
                dist[k]=dist[j]+1
                father[k]=j
```

درخت BFS : درختی است که راس‌های آن، راس‌های گراف بوده و یال‌های آن، یال‌هایی هستند که BFS از طریق آن‌ها انجام شده است. برای مثال در شکل زیر درخت BFS مربوط به گراف سمت چپ، هنگامی که BFS راس ۱ اجرا شود، آمده است:



گراف — درخت BFS

هیچ یالی بین دو سطح با اختلاف ۲ یا بیشتر وجود ندارد.

حال مرتبه زمانی الگوریتم را بررسی می‌کنیم:

– پیاده‌سازی با استفاده از ماتریس مجاورت: n راس داریم که هر کدام یک بار وارد صف می‌شوند و همسایه‌های هر راس هنگام خروج از صف بررسی می‌شود که این فرایند از $\mathcal{O}(n)$ است. پس در این حالت، الگوریتم از مرتبه $\mathcal{O}(n^2)$ می‌باشد.

– پیاده‌سازی با استفاده از لیست مجاورت : n راس داریم که هر کدام یک بار وارد صف می‌شوند. بررسی همسایه‌های راس i، از $\mathcal{O}(d_i)$ است و در نتیجه این الگوریتم از $\mathcal{O}(n+m)$ می‌باشد. اگر گراف همبند باشد، با توجه به این‌که داریم $m \geq n-1$، الگوریتم از $\mathcal{O}(m)$ می‌شود.

کاربردهای BFS:

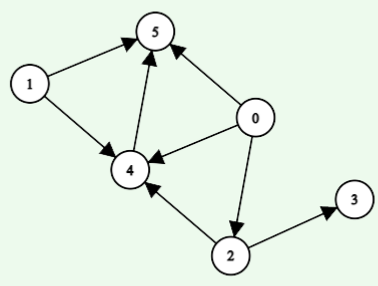
۱. پیدا کردن کوتاه‌ترین مسیر از راس i به راس n در گراف G:
الگوریتم BFS را از راس i اجرا می‌کنیم تا به راس n برسیم. آرایه $dist$ نشان‌دهنده طول مسیر است و از طریق $father$ می‌توانیم مسیر را نیز بیابیم.

۲. گراف دوبخشی:
گراف دوبخشی، گرافی است که بتوان راس‌های آن را به دومجموعه تقسیم کرد به صورتی که بین راس‌های یک بخش یالی نباشد.
قضیه: گراف G دوبخشی است اگر و تنها اگر دوری به طول فرد نداشته باشد.
حال به اثبات قضیه بالا می‌پردازیم:

– با یک مثال ساده (گراف پنج‌راسی ۲ـ منتظم) مشاهده می‌کنیم که اگر گراف دور فرد داشته باشد، دوبخشی نیست.

– می‌خواهیم ثابت کنیم که اگر گراف دور فرد نداشته باشد، دوبخشی است: اگر درخت BFS گراف را رسم کنیم و یک‌درمیان طبقات را با دو رنگ آبی و قرمز رنگ کنیم؛ بین سطح‌های هم‌رنگ قطعا یالی نداریم زیرا فاصله آن‌ها بیشتر یا مساوی ۲ است. بین راس‌های یک‌طبقه نیز یالی نداریم زیرا در این‌صورت بین آن دو راس و پایین‌ترین جد مشترک آن‌ها دوری به طول فرد به وجود می‌آید. بنابراین اگر گراف دوری به طول فرد نداشته باشد، راس‌های آن را می‌توان با دو رنگ به گونه‌ای رنگ کرد که بین راس‌های همرنگ یالی وجود نداشته باشد، پس گراف دوبخشی است.

- گراف جهت‌دار: گرافی است که یال‌های آن جهت دارند.



بعضی از تعاریف مربوط به گراف جهت‌دار به صورت زیر می‌باشد:

– درجه ورودی: تعداد یال‌هایی که به راس وارد می‌شود. $d_{in}(i)$

– درجه خروجی: تعداد یال‌هایی که از راس خارج می‌شود. $d_{out}(i)$

– مولفه‌های قویا همبند: ۲ راس i و j در یک مولفه قویا همبند هستند ، اگر از راس i به j و از j به i مسیر جهت‌دار وجود داشته باشد.

در گراف‌های جهت‌دار، جمع درجات ورودی و جمع درجات خروجی و تعداد یال‌ها برابر هستند و ماتریس مجاورت آن‌ها متقارن نیست.