ساختمان داده‌ها و الگوریتم‌ها

نیم‌سال اول ۰۱ــ۰۰

مدرس: مسعود صدیقین

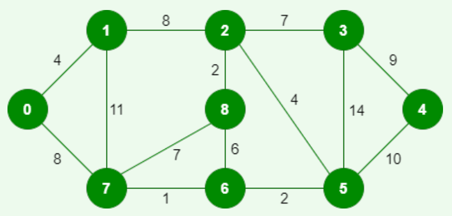
دانشکده‌ی مهندسی کامپیوتر

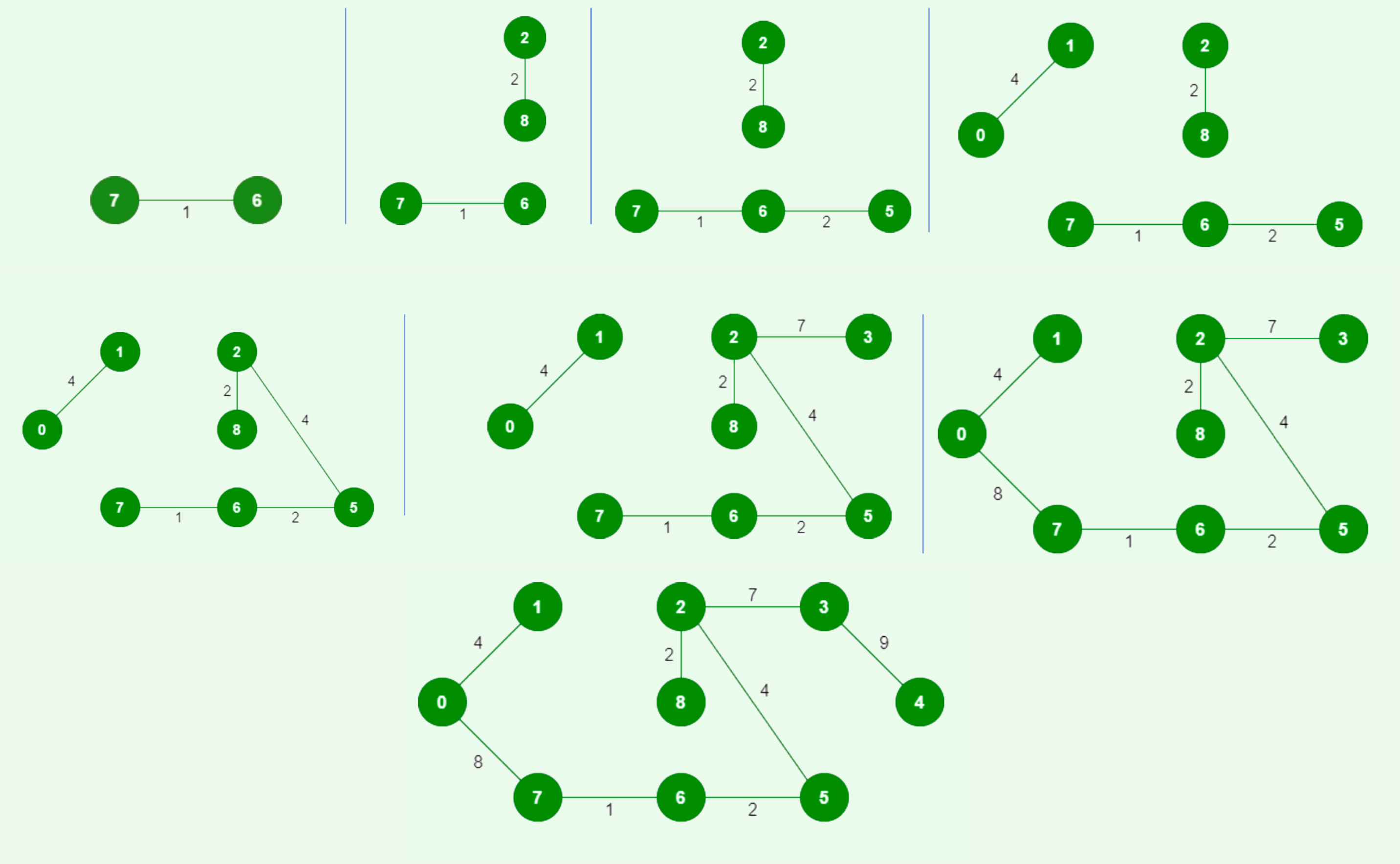
یادآوری جلسه بیست‌وهفتم | **الگوریتم کراسکال** | رامین فراهانی

الگوریتم کراسکال

در جلسه گذشته در مورد الگوریتم کراسکال برای یافتن درخت پوشای کمینه در گراف وزن‌دار صحبت کردیم. همچنین درستی این الگوریتم را اثبات کردیم و مرتبه زمانی آن را مورد بررسی قرار دادیم.

الگوریتم کراسکال: روش کار این الگوریتم به این صورت است که در هر مرحله، کوچک ترین یالی که اضافه کردن آن به مجموعه یال های انتخاب شده، ایجاد دور نمی‌کند را به مجموعه اضافه می‌کند و این کار را تا اضافه شدن $n-۱$ یال ادامه می‌دهد. در شکل‌های زیر، مراحل اجرای این الگوریتم برای گراف زیر آمده است:





درستی الگوریتم کراسکال: با استفاده از برهان خلف درستی این الگوریتم را ثابت می‌کنیم. فرض می‌کنیم OPT پاسخ بهینه و ALG درخت به دست آمده به کمک الگوریتم کراسکال بوده و $OPT \neq ALG$ می‌باشد. یال‌های انتخاب شده در هر دو درخت OPT و ALG را به ترتیب وزن به‌صورت صعودی مرتب می‌کنیم:

$$OPT : e_۱, e_۲, e_۳, ..., e_{n-۱}$$

$$ALG : e'_۱, e'_۲, e'_۳, ..., e'_{n-۱}$$

از آنجا که $OPT \neq ALG$ است، بنابراین اندیس i وجود دارد که $e_i \neq e'_i$. اولین اندیس i که $e_i \neq e'_i$ است را در نظر می‌گیریم. از آنجا که الگوریتم کراسکال در هر مرحله یال با وزن کمینه را انتخاب می‌کند، پس $w(e_i) < w(e'_i)$ می‌باشد. یال e'_i را به درخت OPT اضافه می‌کنیم و در اثر اضافه کردن این یال یک دور ایجاد می‌شود. از آنجا که $e_۱$ تا $e_{i-۱}$ در ALG نیز وجود داشته و با e'_i دور ایجاد نمی‌کنند، پس حتما یک یال e_j در این دور وجود دارد که $j \geq i$ بوده و $w(e_j) > w(e'_i)$ است. با حذف این یال e_j و اضافه کردن یال e'_i وزن درخت OPT به اندازه $w(e_j) - w(e'_i)$ کاهش می‌یابد که با بهینه بودن OPT در تناقض است. پس الگوریتم کراسکال پاسخ بهینه را پیدا می‌کند.

سوال اصلی در این الگوریتم، این است که چگونه بررسی کنیم که آیا با اضافه کردن یک یال، دور ایجاد می‌شود یا خیر. برای این کار ابتدا هر کدام از راس‌ها را در یک مولفه جدا قرار می‌دهیم و برای هر یال مولفه همبندی دو سر آن را چک می‌کنیم($find$)، یک یال را تنها در صورتی اضافه می‌کنیم که دو مولفه همبندی متفاوت را به یکدیگر متصل کند. پس از اضافه کردن هر یال نیز باید مولفه‌های مربوط به دو سر آن یال با یکدیگر ادغام شوند($union$).

برای این کار، به هر راس یک برچسب نسبت می‌دهیم به‌طوری‌که برچسب راس‌های حاضر در یک مولفه یکسان باشد. ابتدا روشی معرفی کردیم که $find$ را در زمان $O(۱)$ و $union$ را در زمان $O(n)$ انجام می‌داد. در این صورت، یال‌ها را در مرتبه $O(mlogm)$ مرتب کرده و با اضافه شدن $n-۱$ یال، $n-۱$ بار $union$ انجام می‌شود و مرتبه کلی الگوریتم $O(mlogm + n^۲)$ خواهد بود. داده ساختاری به نام $disjointset$ معرفی کردیم که عملیات‌های $find$ و $union$ آن به صورت سرشکن در زمان $O(log^*m)$ اجرا می‌شوند. در این‌صورت، زمان کلی الگوریتم از مرتبه $O(mlogm + mlog^*m + nlog^*m)$ می‌شود.

$$log * ۲^{۲^۲} = ۳ \qquad log^* ۲^{۲^{۲^۲}} = log^* ۶۵۵۳۶ = ۴$$