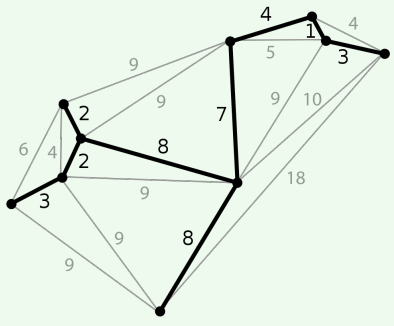
یادآوری جلسه بیست‌وششم — درخت پوشای کمینه — هیربد بهنام

در جلسه‌ی قبل در مورد درخت پوشای کمینه بحث کردیم و الگوریتم پریم برای پیدا کردن درخت پوشای کمینه مربوط به یک گراف را مورد بررسی قرار دادیم. ابتدا با زیردرخت فراگیر و درخت پوشای کمینه آشنا می‌شویم.

زیردرخت فراگیر یک گراف: زیرگرافی شامل همه‌ی راس‌های آن گراف است که درخت باشد.

درخت پوشای کمینه: برای گراف وزن‌دار و همبند G درخت پوشای کمینه، زیردرخت فراگیری است که کمترین جمع وزن یال را داشته باشند. در شکل زیر درخت پوشای کمینه‌ی گراف، به صورت پررنگ مشخص شده است.



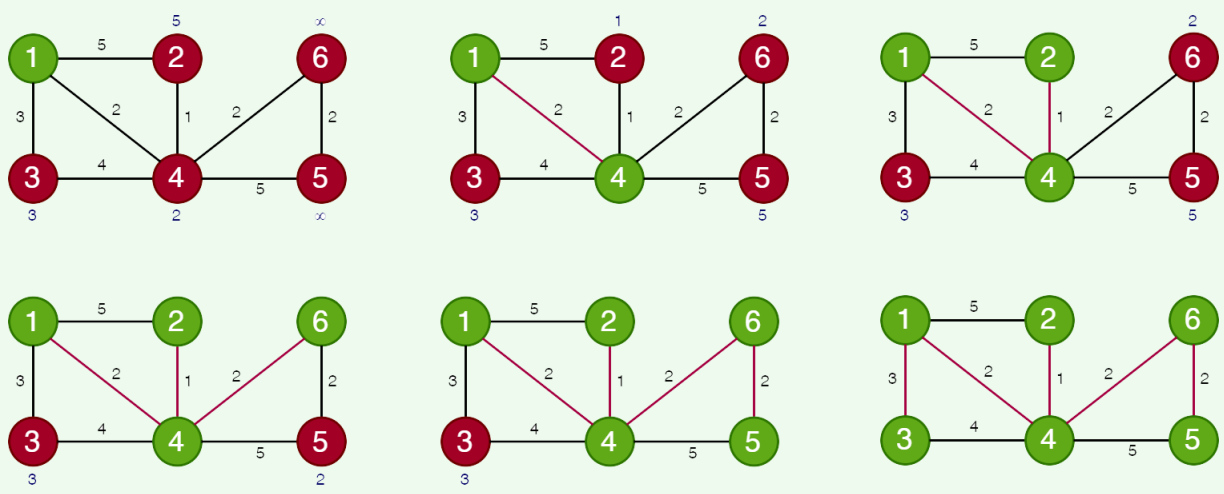
حال الگوریتم پریم را برای به دست آودن درخت پوشای کمینه گراف معرفی می‌کنیم:

الگوریتم پریم: نحوه کار این الگوریتم به این صورت است که آرایه‌ای به نام $dist$ تعریف می‌کنیم و در ابتدا همه‌ی عناصر آن به غیر از همسایه‌های راس شروع، برابر ∞ است؛ برای همسایه‌های راس شروع مقدار $dist$ برابر وزن یال بین راس شروع و آن راس است. همچنین یک مجموعه به نام S در نظر می‌گیریم که نشان‌دهنده‌ی راس‌های مشاهده‌شده است. در ابتدا مجموعه S تنها شامل راس اول بوده و در هر مرحله یک راس به این مجموعه اضافه می‌شود. آرایه‌ای به نام $father$ نیز تعریف می‌کنیم که $father[i]$ نشان‌دهنده‌ی راسی از S می‌باشد که با وزن کمینه به راس i متصل است.

سپس در هر مرحله، راس با کمترین مقدار $dist$ که در V/S وجود دارد را به مجموعه‌ی S اضافه می‌کنیم و یالی را که آن مقدار $dist$ را برای راس مورد نظر ایجاد کرده است، به درخت پوشای کمینه اضافه می‌کنیم و سپس مقدار $dist$ سایر راس‌ها را به‌روز می‌کنیم. این کار را تا جایی ادامه می‌دهیم که V/S تهی شود. شکل زیر شبه کد مربوط به این الگوریتم را نشان می‌دهد:

```
while (S ≠ V)
    select v_j ∈ V/S with min dist
    add v_j to S
    add edge (v_j, father[v_j]) to MST
    for v_k in N(v_j)
        if v_k ∉ S and weight(v_j, v_k) < dist[v_k]
            dist[v_k] = weight(v_j, v_k)
            father[v_k] = v_j
```

شکل زیر تمامی مراحل الگوریتم پریم را برای یافتن درخت پوشای کمینه گراف سمت چپ نشان می‌دهد:



مرتبه زمانی این الگوریتم، در پیاده‌سازی با استفاده از ماتریس مجاورت و لیست مجاورت از $O(n^2)$ است. در پیاده‌سازی به کمک لیست مجاورت، اگر از هرم نیز استفاده کنیم زمان الگوریتم از مرتبه $O(nlogn + mlogn)$ خواهد بود و اگر در این پیاده‌سازی از هرم فیبوناچی کمک گیریم، الگوریتم از مرتبه $O(m + nlogn)$ می‌شود.