درخت کوتاه‌ترین مسیر و ویژگی‌های آن

تعریف

گراف وزن دار، همبند و بدون جهت G را در نظر بگیرید. برای G، یک درخت کوتاه‌ترین مسیر از رأس v، یک زیر درخت فراگیر و ریشه‌دار از راس v مثل T است؛ طوری که به ازای هر رأس u از T، طول مسیر یکتای u, v در درخت، برابر با طول کوتاه‌ترین مسیر بین u, v در G باشد. توجه کنید که به ازای یک G و v، ممکن است بیش از یک درخت کوتاه‌ترین مسیر وجود داشته باشد.

اگر G دور با وزن منفی داشته باشد، نمی‌توانیم هیچ درخت کوتاه‌ترین مسیری برای هیچ‌کدام از راس‌های آن پیدا کنیم (چرا؟). اما اگر G دور با وزن منفی نداشته باشد، به ازای هر رأس G، حداقل یک درخت کوتاه‌ترین مسیر وجود دارد.

الگوریتم

برای پیدا کردن درخت کوتاه‌ترین مسیر برای یک ریشه‌ی تعیین شده مثل v، باید ابتدا به ازای هر رأس u، طول کوتاه‌ترین مسیر بین u, v را محاسبه کنیم. این فاصله را $dist[u]$ می‌نامیم. یک یال از a به b با وزن w می‌تواند در درخت وجود داشته باشد اگر

$$dist[b] = dist[a] + w$$

باشد.

برای محاسبه‌کردن $dist[u]$-ها می‌توان از الگوریتم‌هایی مثل الگوریتم دایکسترا و بلمن-فورد استفاده کرد.

یک راه‌کار، پیدا کردن همه‌ی یال‌های ممکن یرای درخت و سپس استفاده از الگوریتم‌های مثل سرچ و سرچ برای یافتن یک درخت پوشا است.

راه‌کاری دیگر، تعیین کردن پدر هر رأس مثل u در درخت ($par[u]$)، هنگام اجرا کردن الگوریتم دایکسترا یا بلمن-فورد است؛ به این صورت که در ابتدا به ازای همه‌ی u-ها $par[u] = -1$ و $dist[u] = \infty$ قرار می‌دهیم. سپس هرگاه مقدار جدید برای $dist[u]$ پیدا کردیم، $par[u]$ را برابر با رأسی قرار می‌دهیم که مقدار $dist[u]$ را از روی آن آپدیت کرده‌ایم.

در حالت خاص، وقتی وزن همه‌ی یال‌های گراف، واحد باشد (یا به عبارت دیگر گراف وزن‌دار نباشد)، درخت bfs با ریشه‌ی v، یک درخت کوتاه‌ترین مسیر برای v است.

پیچیدگی الگوریتم

بسته به استفاده کردن از الگوریتم‌های دایکسترا و بلمن-فورد، پیچیدگی الگوریتم نیز مانند آن‌ها می‌شود.

شبه کد با پیاده‌سازی دایکسترا

```
Function Dijkstra(v):
        dist[v] = 0

        For each vertex u in Graph:
                If u != v
                        dist[u] = infinity
                        par[u] = -1
                End if
```

```
        End for

        While T != 0 :
                u := vertex not in T with min dist[u]
                Add u to T

                For each neighbor k of u:
                        If(dist[k] != dist[u]+w[u][k]):
                                dist[k] = dist[u]+w[u][k]
                                par[k] = u
                End for
        End while
End Function
```

شبه کد با پیاده‌سازی بلمن-فورد

```
Function BellmanFord(list vertices, list edges, vertex v)
        // Step 1: initialize graph
        For each vertex u in vertices:
                If u ≠ v :
                        dist[u] = infinty
                        par[u] = -1
                End if
        End for
        // Step 2: relax edges repeatedly
        For i from 1 to size(vertices) − 1:
                For each edge (u, k) with weight w in edges:
                        If dist[u] + w < dist[k]:
                                dist[k] = dist[u] + w
                                par[k] = u
                        End if
                End for
        End for
End Function
```

مسائل نمونه

مراجع