الگوریتم بلمن-فورد

تعریف

الگوریتم بلمن-فورد راه‌کاری برای پیدا کردن کم‌وزن مسیر از رأس مشخص آغاز به بقیه رئوس در گراف جهت‌دار و وزن‌دار می‌دهد. وزن یک مسیر در گراف وزن‌دار برابر مجموع وزن یال‌های آن است. جهت‌دار نبودن یال‌ها هم مشکلی ایجاد نمی‌کند و می‌توان برای یال‌های غیر جهت‌دار دو یال فرض کرد.

یکی از مزیت های این الگوریتم نسبت به الگوریتم دایکسترا توانایی اجرا شدن روی گراف‌ها با یال منفی است.

کاربردها

این الگوریتم علاوه بر پیدا کردن کوتاه‌ترین مسیر به پیدا کردن دور منفی در گراف‌ها کمک می‌کند. بنابراین استفاده‌های بیشتری از الگوریتم‌های مشابه می‌تواند داشته باشد.

الگوریتم

ابتدا روش کار این الگوریتم را بررسی می‌کنیم و سپس درستی آن را بررسی می‌کنیم. فرض کنید $1 \leq s \leq n$ که در آن رأس s رأس آغاز است و فرض کنید:

$$dist(s) = 0$$

و به ازای هر $v \neq s$:

$$dist(v) = \infty$$

این الگوریتم به تعداد یکی کمتر از رأس ها در هر مرحله روی همه‌ی یال‌ها عملیات زیر را انجام می‌دهد:

$$dist(u) = min\Big(dist(u), dist(v) + w(v,u)\Big)$$

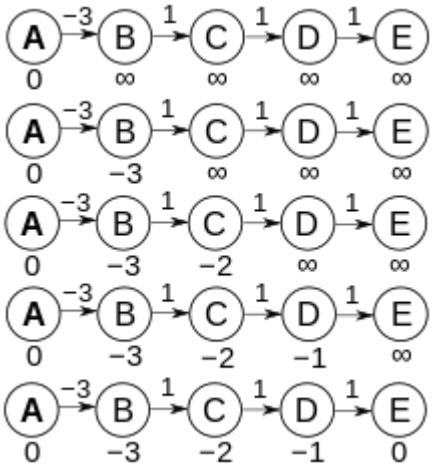
در واقع فاصله دو سر یال را با توجه به وزن آن تصحیح می‌کند. به این عملیات در اصطلاح $Relax$ کردن یال ها می گویند.

اثبات درستی

درستی این الگوریتم را به استقرا ثابت می‌کنیم. فرض کنید این الگوریتم تا i-امین بار اجرا شدن کوتاه‌ترین فاصله تمامی رئوسی که کم‌وزن‌ترین مسیر آن‌ها حداکثر i یال دارد را پیدا کند. پایه استقرا: در مرحله صفر-ام رأس آغاز فاصله‌اش صفر است؛ پس درست است. گام استقرا: هر یک از رأس‌هایی که کوتاه‌ترین مسیرشان حداکثر $i+1$ یال دارد آخرین یالشان حتما به یک رأسی است که در مرحله قبلی فاصله‌شان پیدا شده است(در واقع حداکثر از i یال استفاده می‌کنند). پس بعد از $Relax$ کردن یال ها برای بار $i+1$-ام جواب تمامی رأس‌هایی که کوتاه‌ترین مسیرشان $i+1$ یالی است را پیدا کرده‌ایم. پس درستی الگوریتم ثابت می‌شود.

یک مثال

در گراف زیر، روند اجرای این الگوریتم را می‌توانید مشاهده کنید



پیچیدگی الگوریتم

به ازای هر رأس باید روند بالا را طی کنیم. یعنی دنبال آن بگردیم و همه‌ی همسایه‌های آن را پیمایش کنیم. پس پیچیدگی زمانی برنامه از $O(n \times n) = O(n^2)$ است. هر چند می‌توان با استفاده از داده ساختار هرم پیاده‌سازی از $O\left(e \times lg(n)\right)$ ارائه داد.

پیاده‌سازی اولیه

شبه کد:

```
  function BellmanFord(list vertices, list edges, vertex s)

   // Step 1: initialize graph
   for each vertex v in vertices:
       if v ≠ s then dist[v] = INF

   // Step 2: relax edges repeatedly
   for i from 1 to size(vertices)-1:
       for each edge (u, v) with weight w in edges:
           if dist[u] + w < dist[v]:
               dist[v] = dist[u] + w
```

پیاده‌سازی

```
#include <iostream>
#include <vector>

using namespace std;

typedef pair<int, int> pii;
const int MAXN = 1000*100 + 10;
const int INF = 1000*1000*1000;
int dist[MAXN];
vector<pii> edge;
vector<int> w;
int n, m;

void readInput(){
        cin >> n >> m;
        for(int i=0; i<m; ++i){
                int u, v, z;
                cin >> u >> v >> z;
                edge.push_back(pii(u, v));
                w.push_back(z);
        }
}
```

```
void bellmanFord(int s)
{
        dist[s] = 0;
        for(int i=0; i<n; ++i)
                if(i!=s)
                        dist[i] = INF;
        for(int i=0; i<n-1; ++i)
                for(int j=0; j<(int)edge.size(); ++j){
                        int u = edge[j].first;
                        int v = edge[j].second;
                        dist[v] = min(dist[v], dist[u]+w[j]);
                }
}

void writeOutput()
{
        for(int i=0; i<n; ++i)
                cout << dist[i] << " ";
        cout << endl;
}

int main(){
        readInput();
        bellmanFord(0);
        writeOutput();
        return 0;
}
```

مسائل نمونه

۱. مسئله‌ی ۲۳۶ سایت ۲۳۶ [illegible]

مراجع

۱. [illegible]