Dijkstra

Autor: Franciszek Pietrusiak

Algorytm Dijkstry

Standardowy Problem

Dany jest **ważony** graf G (wagi są nieujemne) z n wierzchołkami i m krawędziami. Mamy wyróżniony jeden wierzchołek startowy s. Chcemy dla wszystkich innych wierzchołków w G odpowiedzieć na pytanie: Jaka jest najkrótsza ścieżka idąca z s do tego wierzchołka.

Algorytm Dijkstry

Dla każdego wierzchołka będziemy w tablicy d trzymali długość najkrótszej ścieżki z s do tego wierzchołka oraz to, czy był on odwiedzony. Początkowo $d[s] = 0$ oraz $d[v] = \infty$, gdzie $v \neq s$, a ∞ oznacza dostatecznie dużą liczbę. Algorytm Dijkstry wykonuje n iteracji. W każdej iteracji przeglądany jest **jeszcze nie odwiedzony** wierzchołek v, którego $d[v]$ jest najmniejsze. Zaznaczamy go jako odwiedzonego. Następnie wykonywane są *relaksacje* krawędzi: Załóżmy, że sąsiadem v jest wierzchołek u, a waga krawędzi $v \to u$ to w. Jeśli zachodzi, że $d[v] + w < d[u]$, to znaczy, że poprawiliśmy rozwiązanie dla u. Po próbach relaksacji krawędzi wychodzących z v iteracja się kończy. Po odwiedzeniu wszystkich wierzchołków algorytm się kończy, a tablica d przechowuje poprawne odpowiedzi dla każdego wierzchołka w G.

Dowód

Algorytm bazuje na twierdzeniu, że jeśli wierzchołek zostaje odwiedzony, to jego d jest poprawne i nie ulega zmianie.
Można to udowodnić indukcyjnie po liczbie wierzchołków odwiedzonych. Na samym początku odwiedzony jest tylko s oraz $d[s] = 0$ i jest to najlepszy możliwy wynik. Załóżmy teraz indukcyjnie, że teza zachodzi dla wszystkich wcześniejszych iteracji. W obecnej iteracji wybieramy wierzchołek v. Chcemy udowodnić, że $d[v]$ jest równe długości najkrótszej ścieżki z s do v, którą oznaczymy $l[v]$. Jeśli z s nie istnieje żadna ścieżka do v to równość zachodzi. W przeciwnym wypadku ścieżka ta składa się z dwóch części: pierwsza część składa się tylko z wierzchołków już odwiedzonych. Ostatni z nich oznaczmy jako q. Druga cześć zaczyna się z wierzchołka nieodwiedzonego p i może przebiegać przez odwiedzone wierzchołki aż dotrze do v. Zauważmy, że q i p są połączone krawędzią $q \to p$. Zacznijmy od udowodnienia, że $d[p] = l[p]$. Jest tak dlatego, że z założenia indukcyjnego we wcześniejszej iteracji wybraliśmy q. Zachodzi $d[q] = l[q]$, a skoro p leży na najkrótszej ścieżce, to podczas relaksacji krawędzi $q \to p$ nastąpiło ustawienie $d[p] = l[p]$.
Teraz zajmijmy się wierzchołkiem v. Wiadomo, że zawsze zachodzi $d[v] \geq l[v]$. Dodatkowo skoro wagi G są nieujemne, to $l[p] \leq l[v]$. Jednakże skoro p i v są nieodwiedzone, a v zostało wybrane przed p, to zachodzi $d[v] \leq d[p]$. Zatem zachodzi

$$d[p] = l[p] \leq l[v] \leq d[v] \leq d[p].$$

Więc $d[v] = l[v]$ co kończy dowód.

Implementacja

Implementacja z użyciem kolejki priorytetowej:

```
for (int i=1; i<=n; i++)
                d[i] = INF;

d[1] = 0; // 1 jest wierzchołkiem startowym s
priority_queue<pair<ll, int>, vector<pair<ll, int>>, greater<pair<ll, int>>> Q;
Q.push({0, 1});

while (!Q.empty()) {
        int v = Q.top().SD;
        ll dist = Q.top().FR;
        Q.pop();

        if (dist != d[v]) // ważne! Nie chcemy sprawdzać tego samego wierzchołka wiele razy
                continue;

        for (auto [u, w] : G[v])
                if (d[u] > dist + w) {
                        d[u] = d[v] + w;
                        Q.push({d[u], u});
                }
}
```

Przy takiej implementacji złożoność wynosi $O(m \log n)$.

Zadanie Przemytnicy

Zadanie Wielka Ucieczka Traktorem

Zadanie Zawody

Źródła

- https://cp-algorithms.com/graph/dijkstra.html
- https://cp-algorithms.com/graph/dijkstra_sparse.html