

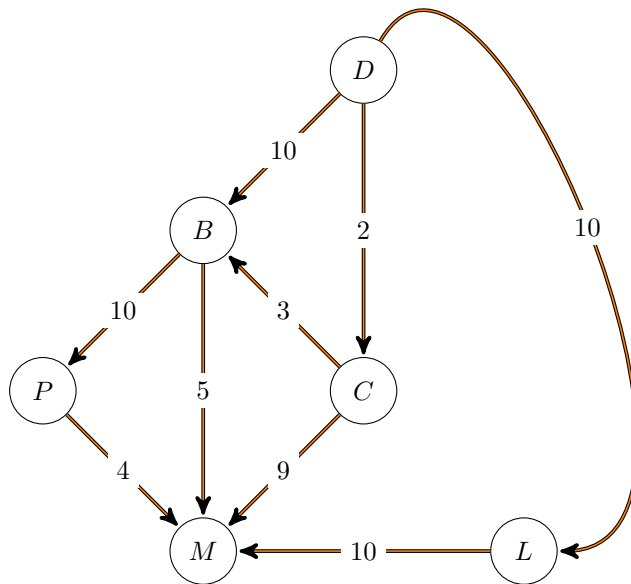
Lista 6

Zadanie 1

1 Problem

Rozważamy skierowany graf acykliczny (DAG) $G = (V, E, f)$ gdzie każda krawędź $e \in E$ ma swój koszt określany przez funkcję $f : V \times V \rightarrow \mathbb{R}$. Chcemy znaleźć wszystkie najkrótsze („najtańsze”) ścieżki od źródła ($v_s \in V$ t. że $\deg^-(v_s) = 0$) do pozostałych wierzchołków $V \setminus \{v_s\}$.

W przykładzie poniżej wierzchołkiem źródłowym jest wierzchołek D .



Rysunek 1: Przykładowy DAG

2 Concept

Użyjemy tutaj metodologii programowania dynamicznego. Zapamiętujemy każdą nowo odkrytą najkrótszą ścieżkę pomiędzy danymi wierzchołkami, a podczas dalszego znajdowania ścieżek wykorzystujemy tę wiedzę, którą już mamy.

3 Rozwiązanie

Najpierw musimy ustalić kolejność, według której będziemy przeglądać wierzchołki. Wierzchołki grafu wejściowego sortujemy topologicznie względem wierzchołka startowego. Użyjemy tutaj algorytmu Kahn'a:

Algorithm 1 Algorytm Kahn'a

```
1:  $L \leftarrow []$ 
2:  $S \leftarrow \{s\}$ 
3: while  $S \neq \emptyset$  do:
4:   remove a node  $n$  from  $S$ 
5:    $L \leftarrow \text{concat}(L, [n])$ 
6:   for each node  $m$  with an edge  $e$  from  $n$  to  $m$  do:
7:     remove edge  $e$  from the graph
8:     if  $\text{deg}^-(m) = 0$  then:
9:       insert  $m$  into  $S$ 
10:    end if
11:  end for
12: end while
```

Złożoność obliczeniowa powyższego algorytmu wynosi $O(|V| + |E|)$. Algorytm Kahn'a daje nam „zlinearyzowany” graf, po którym możemy szukać najkrótszych ścieżek pomiędzy v_s a pozostałymi wierzchołkami.

Zdefiniujmy funkcję zwracającą nam indeksy wszystkich wierzchołków sąsiednich dla danego $v_k \in V$:

$$\mathcal{N}(k) = \{j : \forall (v_j, v_k) \in E\}.$$

Nasze poszukiwania zaczynamy od inicjalizacji tablicy odległości pomiędzy wszystkimi wierzchołkami V a wierzchołkiem v_s . Tablicę odległości $d[0, \dots, |V|]$ wypełniamy wartościami ∞ , przy czym $d[s] = 0$. Wówczas $\forall_{i \in \{0, \dots, |V|\}}$ $d[i]$ określa odległość pomiędzy wierzchołkiem źródłowym v_s a wierzchołkiem v_i .

Oprócz tablicy d musimy przechowywać numery indeksów wierzchołków, które tworzą znaną ścieżkę z v_s do v_i . Stworzymy listę list, którą nazwiemy r , gdzie $r[i]$ to lista indeksów wierzchołków, które tworzą najkrótszą ścieżkę pomiędzy v_s a v_i .

```
1:  $d \leftarrow [\infty, \dots, \infty]$ 
2:  $d[s] \leftarrow 0$ 
3:  $r \leftarrow [ [], \dots, [] ]$ 
4: for each  $i$  in  $\{i : v_i \in V\}$  in linearized order do:
5:   for each  $j$  in  $\mathcal{N}(i)$  do:
6:     if  $d[i] > d[j] + f(j, i)$  then:
7:        $d[i] \leftarrow d[j] + f(j, i)$ 
8:        $r[i] \leftarrow \text{concat}(r[j], [i])$ 
9:     end if
10:  end for
11: end for
```

W linijkach 6-9 sprawdzamy, czy nie ma „tańszej” drogi z wierzchołka v_s do wierzchołka v_i . Złożoność obliczeniowa powyższego algorytmu wynosi $O(|E|)$, ponieważ w pętli rozważamy koszty powstałe dla każdej krawędzi w grafie.