Zadanie 6.
W zadaniu 6. należało wyznaczyć najkrótszą ścieżkę między wierzchołkami s i d w skierowany grafie z dopuszczalnymi wagami ujemnymi. Zakładamy, że najkrótsza ścieżka pomiędzy dowolnymi dwoma wierzchołkami przechodzi przez co najwyżej k krawędzi. Nie zakładam jednak, że między wszystkimi wierzchołkami muszą występować najkrótsze ścieżki (czyli mogą istnieć cykle o wadze ujemnej). Wtedy dla wierzchołków, między którymi nie istnieje najkrótsza ścieżka, zostanie wypisana dotychczas znaleziona najkrótsza między nimi, czyli nie dłuższa niż k kroków.
Procedura $FindPath$ przyjmuje parametry s (źródło), d (cel), E (zbiór krawędzi postaci (u, v, w) - wierzchołki u i v, waga w), n (liczba wierzchołków), k (maksymalna liczba kroków).

Algorithm 1 Najkrótsza ścieżka

procedure FindPath(s,d,E,n,k)
    $answers \leftarrow []$
    $distance \leftarrow [\infty \cdot n]$ //tablica n-elementowa wypełniona ∞
    $distance[s] \leftarrow 0$
    $parents \leftarrow [[] \cdot n]$ //tablica n-elementowa wypełniona pustymi tablicami
    for $i \leftarrow 0$ to $k-1$, i++ **do**
        $currentdistance \leftarrow [\infty \cdot n]$
        for $edge$ in E **do**
            if $currentdistance[edge[1]] > distance[edge[0]] + edge[2]$ **then**
                $currentdistance[edge[1]] \leftarrow distance[edge[0]] + edge[2]$
                $parents[edge[1]].add([edge[0], i+1])$
                if $edge[1] = d$ **then**
                    $answers.add([currentdistance[edge[1]], i+1])$
        $distance \leftarrow currentdistance$
    return min(answers), parents

Zewnętrzna pętla for wykonuje się k razy, wewnętrzna iteruje po wszystkich krawędziach w E. Operacje w pętlach mają stałą złożoność obliczeniową, natomiast wyznaczenie minimum z $ansewers$ (zakładam, że jest to minimalny koszt drogi, stąd porównywane elementy to answers[i][0] $\forall i \in 0, ..., size(answers)$) jest rzędu $O(k)$, ponieważ maksymalnie może tam być k elementów. Całkowita złożoność to zatem $O(k \cdot |E| + k) = O(k \cdot (|E| + 1)) = O(k \cdot |E|)$.

Dla pary wierzchołków, która nie posiada najkrótszej trasy (przez cykle ujemne) zostanie wyznaczona trasa dotychczas najkrótsza (czyli składająca się maksymalnie z k kroków). Dla wierzchołków, między którymi najkrótsza trasa istnieje, będzie to właśnie ta trasa. W tym celu każdą znalezioną trasę, która prowadzi mnie do d trzymam w $answers$. Po wyznaczeniu tej, która ma najmniejszą długość otrzymuję również całkowitą liczbę wykonanych kroków. Trzymając w $parents$ wierzchołek oraz krok w którym do niego dotarłem, jestem w stanie odtworzyć trasę za pomocą algorytmu znajdującego się niżej.

Do odtworzenia trasy niezbędne są zwracane przez FindPath wartości, czyli koszt pokonanej trasy wraz z ilością kroków oraz rodzice poszczególnych wierzchołków wraz z odpowiednim krokiem, który został wykonany aby do nich dojść. Zakładając, że mogą istnieć cykle nie wystarczyło trzymać jednego rodzica, ponieważ na wejściu do cyklu po jednorazowym jego przejściu informacja o pierwotnym rodzicu zostaje nadpisana. Zamiast tego trasa będzie odtwarzana na podstawie kolejnych zapisanych kroków oraz odpowiadających im wierzchołkom z listy *parents*, startując od celu i wracając do źródła.

Algorithm 2 Odtwarzanie ścieżki

```
procedure ShowPath(answer, parents, d)
    cost ← answer[0]
    k ← answer[1]
    write(cost) //wypisywanie kosztu trasy
    while k > 0 do
        for i ← size(parents[d])-1 to i- -,  do
            if parents[d][i][1] = k then
                write(parents[d][i][0]) //wypisywanie k-i wierzchołka na trasie
                d ← parents[d][i][0]
                k ← k-1
                break
```

Pierwotnie zostanie wypisany koszt trasy, następnie odtworzona trasa wstecz.