```
def dijkstra(G):
    vis = [False] * n
    q = PrioQ()
    while len(q) > 0:
        d, v = q.pop()
        for u in G(v):
            if not vis[u]:
                q.put((d + w(v,u), u))
                vis[u] = True
```

Zadanie 1: Windy w drapaczu chmur

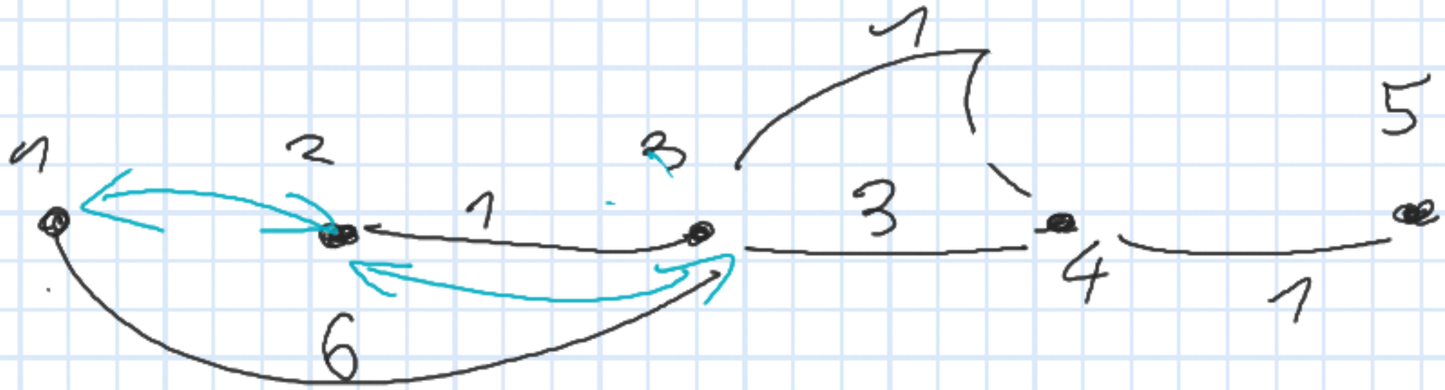
Wieżowiec ma 100 pięter i n wind, nie ma natomiast schodów. Każda winda posiada listę pięter, do których dojeżdża i prędkość w sekundach na piętro.

Jesteśmy na piętrze i, chcemy się dostać na piętro j. Ile minimalnie sekund musimy spędzić w windach, aby tam dotrzeć?

w1: [1,3,4] 3

w2: [2,3,4,5] 1

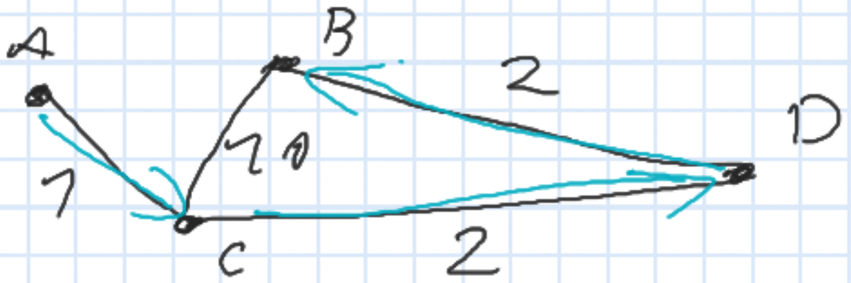
w3: {1,2,3}

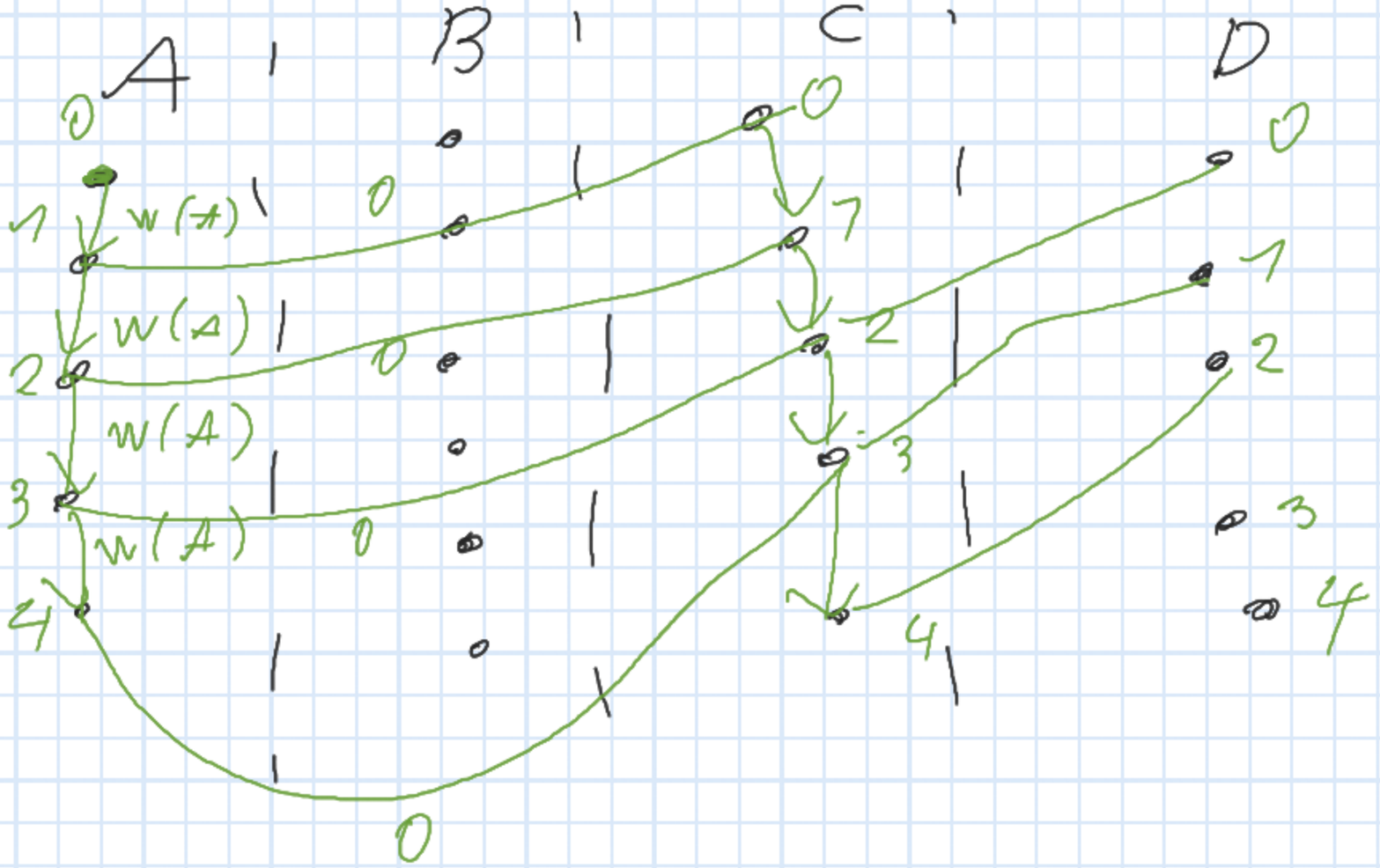


Dostajemy na wejściu graf, w którym wierzchołkami są miasta, a krawędziami drogi między nimi. Dla każdego miasta znamy cenę paliwa w złotych na litr, a dla każdej drogi jej długość w kilometrach.

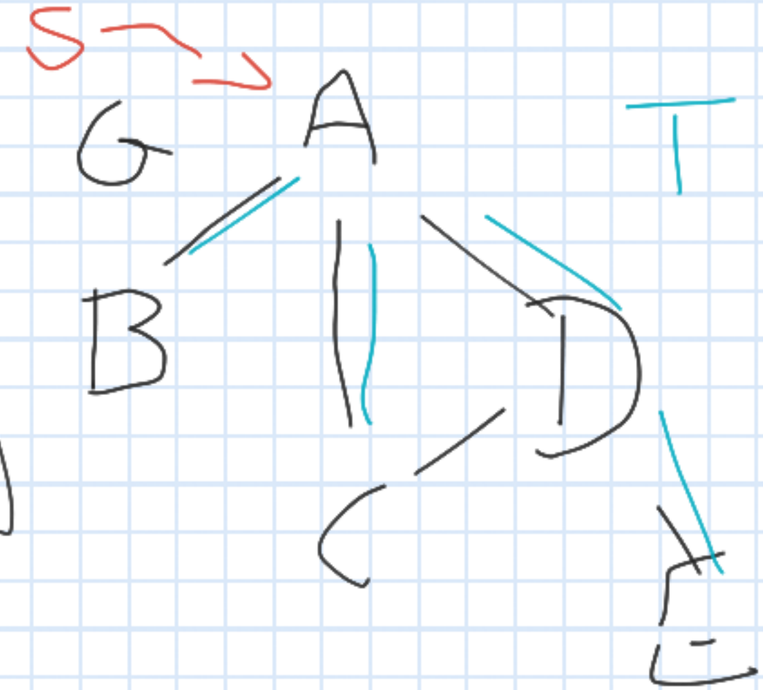
Nasz samochód ma zbiornik pojemności 100 litrów i pali jeden litr na kilometr.

Startujemy z miasta A z pustym zbiornikiem. Ile minimalnie musimy zapłacić za paliwo, aby dotrzeć do miasta B?





Dany jest graf ważony G, oraz drzewo rozpinające T zawierające wierzchołek s. Podaj algorytm, który sprawdzi, czy T jest drzewem najkrótszych ścieżek od wierzchołka s.



$[(A-B), (A-C), (A-D), D-E]$

```
r(e):
    if f, t = e
        d(f) + w(e) > d(t)
            ret False
```

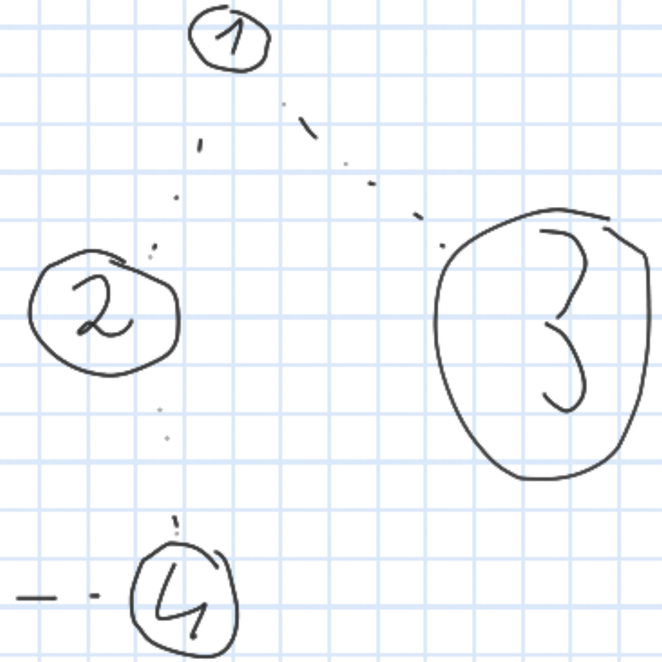
Dostajemy na wejściu listę trójek (miastoA, miastoB, koszt). Każda z nich oznacza, że możemy zbudować drogę między miastem A i B za podany koszt. Ponadto, w dowolnym mieście możemy zbudować lotnisko za koszt K, niezależny od miasta. Na początku w żadnym mieście nie ma lotniska, podobnie między żadnymi dwoma miastami nie ma wybudowanej drogi.

Naszym celem jest zbudować lotniska i drogi za minimalny łączny koszt, tak aby każde miasto miało dostęp do lotniska.

Miasto ma dostęp do lotniska, jeśli:

1) jest w nim lotnisko, lub
2) można z niego dojechać do innego miasta, w którym jest lotnisko

Jeżeli istnieje więcej niż jedno rozwiązanie o minimalnym koszcie, należy wybrać to z największą ilością lotnisk.



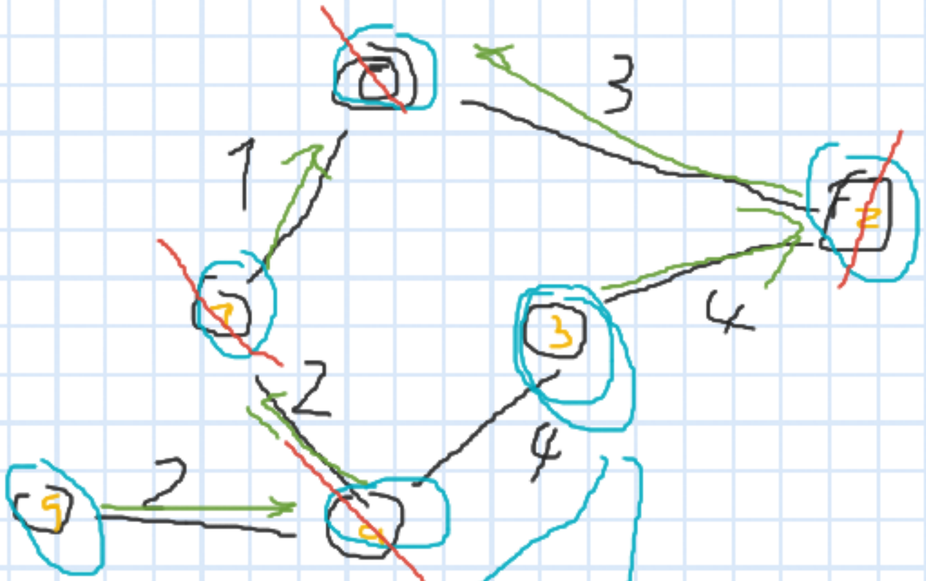
[(1,2,8), (1,3,10),
(2,4,6), (4,5,7)]

K = 9

Dany jest graf ważony G. Ścieżka superfajna, to taka, która jest nie tylko najkrótszą wagowo ścieżką między v i u, ale także ma najmniejszą liczbę krawędzi (inaczej mówiąc, szukamy najkrótszych ścieżek w sensie liczby krawędzi wśród najkrótszych ścieżek w sensie wagowym). Podaj algorytm, który dla danego wierzchołka startowego s znajdzie superfajne ścieżki do pozostałych wierzchołków.

(dist, v)

(d_w, d_v, v)



V (1, 1, 1)

V (3, 1, 1)

V (3, 2, 4)

V (7, 2, 3)

(5, 3, 5)

(7, 3, 3)

Dostajemy na wejściu trzy stringi: A, B i C. A i B są tej samej długości. Zachodzą następujące właściwości:

1) Litery na tym samym indeksie w stringach A i B są równoważne
2) Jeżeli litera a jest równoważna z literą b, to litera b jest równoważna z literą a
3) Jeżeli litera a jest równoważna z b, a litera b z literą c, to litera a jest równoważna z literą c
4) Każda litera jest równoważna sama ze sobą

W stringu C możemy zamienić dowolną literę z literą do niej równoważną. Jaki jest najmniejszy leksykograficznie string, który możemy w tej sposób skonstruować?