algograf

Projekt 1: Parada

W ramach projektu należy napisać program w języku Python rozwiązujący poniższe zadanie. Następnie program należy wysłać do oceny poprzez następujący kurs na platformie UPEL:

- nazwa: Algorytmy grafowe 2023/2024 - projekty
- hasło: galgo2023

Projekt zostanie oceniony w następujący sposób:

- osoby z najlepszymi programami (10 najszybszych) otrzymają +1.5 do oceny końcowej
- osoby z dobrymi programami otrzymują +1.0 do oceny końcowej
- osoby z programami niezbyt dobrymi (rozwiązującymi mało testów) otrzymują +0.5 do oceny końcowej
- osoby, które nie nadeślą programu lub ich program nie będzie działał otrzymują +0.0 do oceny końcowej

Termin nadsyłania rozwiązań: **7 stycznia 2024, 23:59**

Treść zadania

Królestwo Bajtycji przygotowuje się do obchodów 100-nej rocznicy zakończenia wieloletniej wojny z sąsiednim Księstwem Qubicji. Z tej okazji Król Bajtoklecjan postanowił zorganizować paradę wojskową ulicami stolicy Królestwa - Bajtogrodu. Twoim zadaniem będzie wyznaczenie trasy parady.

Defilada maszerować będzie przez rozmaite place Bitogrodu, połączone ze sobą ulicami. Wyjątkowe znaczenie dla mieszkańców stolicy mają place nazywawne *tranzytowymi* - są to te place, które leżą na każdej drodze pomiędzy pewnymi dwoma sąsiadującymi z nimi placami. Po wojnie wybudowano na każdym z nich łuk tryumfalny, by uczcić pamięć przechodzących tamtędy, zmierzających na front żołnierzy. Łuki te ustawione są w taki sposób, że aby przez nie przejść, trzeba podążać ścieżką łączącą pewne dwa sąsiadujące place o wspomnianej wyżej własności (tj. aby przejść przez łuk tryumfalny na placu tranzytowym A należy przyjść z sąsiedniego placu B i kierować się do innego sąsiedniego placu C takich, że każda ścieżka między B i C prowadzi przez plac A).

Z uwagi na ich prestiżową rolę, parada musi przejść przez największą możliwą liczbę łuków tryumfalnych. Trasa nie może prowadzić przez żaden z placów więcej niż jeden raz. Z uwagi na

nadchodzącą falę upałów, w trosce o zdrowie swoich żołnierzy, Król pragnie również by czas trwania uroczystości był jak najmniejszy, przy zachowaniu tych warunków.

Dane wejściowe

Do zaimplementowanej funkcji przekazane zostaną następujące argumenty:

- *N* - ilość placów w Bitogrodzie
- lista zawierająca krotki postaci *(a, b, t)*, gdzie *a*, *b* to numery placów, a *t* to czas potrzebny na przejście ulicą łączącą place *a* i *b* Place są indeksowane od *1*, tj. mają indeksy *1*, *2*, ..., *N*.

Przykładowe wywołanie może wyglądać następująco:

```
solve(7, [
  (1, 2, 2),
  (2, 3, 3),
  (3, 4, 5),
  (4, 6, 1),
  (4, 5, 2),
  (4, 7, 3),
])
```

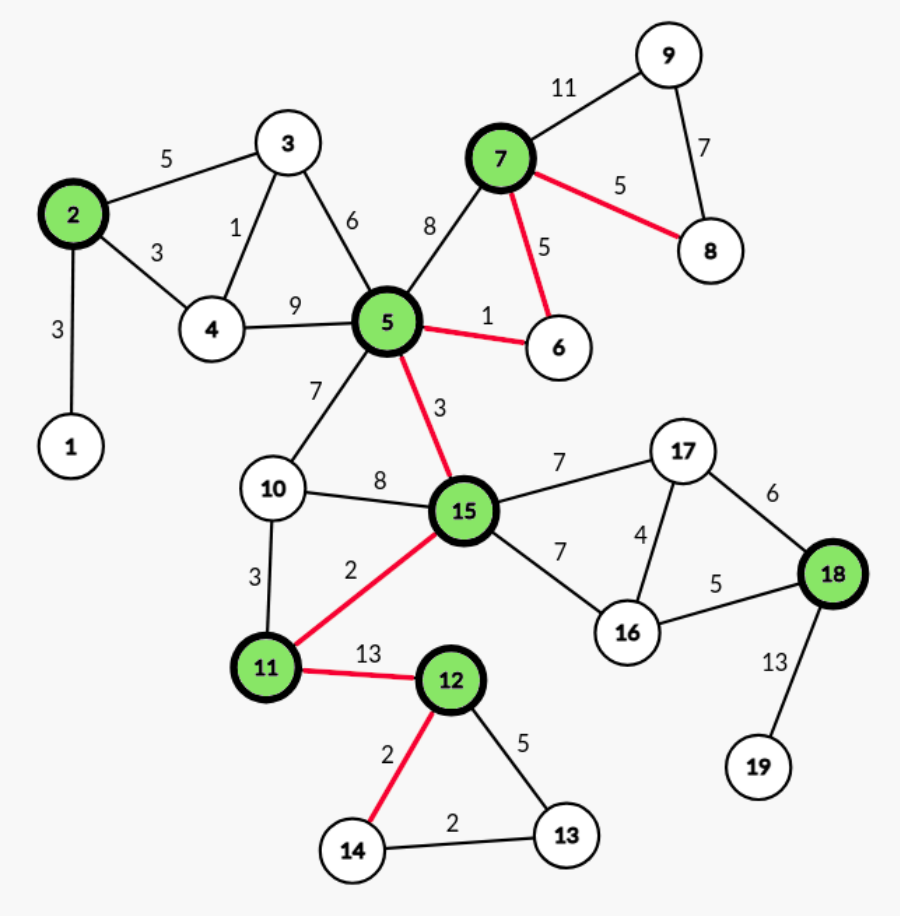
Funkcja *solve* powinna wyznaczać optymalną według opisanych wyżej kryteriów trasę, a następnie zwracać parę liczb - ilość łukótw tryumfalnych, przez które przejdzie parada, oraz całkowity czas jej trwania.

Dla powyższych danych place tranzytowe to *2*, *3* i *4*. Możliwe jest przejście przez wszystkie z nich, a najkrótsza możliwa droga to np. *1*, *2*, *3*, *4*, *6* o łącznym czasie *2 + 3 + 5 + 1 = 11* (aby przejść przez łuk tryumfalny, nie wystarczy wejść na jego plac - nie można zatem pominąć pierwszego i ostatniego placu powyższej trasy).

Można przyjąć, że grafy w zadaniu są spójne i zawierają co najmniej jeden plac tranzytowy.

Przykład

Rozważmy następujący układ placów:



Place zaznaczone na zielono to place tranzytowe. Dla przykładu: plac *15* jest placem tranzytowym, ponieważ każda ścieżka między *10* a *17* przechodzi prez *15*.

Na czerwono zaznaczona została optymalna ścieżka. Podążając nią, parada przejdzie pod czterema łukami tryumfalnymi - *7*, *5*, *11* i *12*. Jakkolwiek plac tranzytowy *15* znajduje się na trasie przemarszu, idąc podaną trasą parada nie przejdzie pod jego łukiem tryumfalnym - przyjdzie do niego z placu *5* i wyjdzie w kierunku placu *11*, a place *5* i *11* poza ścieżką *5*, *15*, *11* łączy również ścieżka *5*, *10*, *11*, a zatem warunek przejścia pod łukiem tryumfalnym placu *15* nie jest spełniony.

Dla takiej instancji problemu prawidłowa odpowiedź to zatem *(4, 31)*.

Instrukcja

Infrastruktura do projektu dostępna jest w formie archiwum z plikami źródłowymi w języku Python (link na dole). Szkielet rozwiązania znajduje się w pliku *example.py* - importuje on funkcję `runtests` z modułu `data` i uruchamia ją, podając swoją funkcję rozwiązującą jako argument. Przesyłane rozwiązania powinny mieć postać analogicznego pliku. Przetestować rozwiązanie można uruchamiając ów plik, np.

```
python3 example.py
```

Na wyjście standardowe wypisane zostaną informacje o rezultatach poszczególnych testów, a także podsumowanie z ilością testów zakończonych sukcesem i przybliżonym łącznym czasie

obliczeń.

Warunki techniczne

- Program powinien być napisany w języku Python i działać z wersją 3.8.10
- Program nie może wykorzystywać zewnętrznych bibliotek (biblioteka standardowa jest dopuszczalna)
- Dopuszczalne jest zwiększenie limitu głębokości rekursji i w razie potrzeby rozmiaru stosu systemowego, np.

```
import sys
sys.setrecursionlimit(100000)

import resource as rc
rc.setrlimit(rc.RLIMIT_STACK, (rc.RLIM_INFINITY, rc.RLIM_INFINITY))
```

Pliki

- project1.zip