algograf

Projekt 1: Królewski Trakt

Treść zadania

W czasach świetności, istotne punkty Królestwa Bajtycji - miasta, wsie, fortece - łączyła gęsta sieć dróg. Czasy świetności minęły, a pustki w skarbcu spowodowały ograniczenie wydatków na infrastrukturę. Dziś dobrze utrzymany pozostał tylko Królewski Trakt - najkrótszy zestaw dróg łączący wszystkie ważne punkty Bajtycji. Ta sytuacja niestety ułatwia zadanie bandytom, którzy coraz częściej i bardziej zuchwale napadają na ludzi nim podróżujących.

Do zapewnenia bezpieczeństwa poddanych, Król Bajtezjan chce wykorzystać swoich wasali. Każdy znaczniejszy lord Bajtycji posiada kilka zamków i fortów, pomiędzy którymi mógłby wysyłać Królewskim Traktem patrole swoich żołnieży. Władca postanowił wybrać kilku z nich i nadać im tytuł Obrońcy Traktu, który zobowiąże ich do dbania o porządek na drogach pomiędzy swoimi fortyfikacjami w zamian za możliwość pobierania myta od podróżujących kupców. Jako, że rola ta wiąże się również z prestiżem i politycznymi wpływami, kandydatów nie brakuje, a i sam król chętnie nagrodziłby w ten sposób jak największą liczbę swoich wiernych lenników. Niestety, lordowie są zachłanni i niezbyt zgodni, więc chcieliby mieć każdy fragment Królewskiego Traktu pomiędzy swoimi fortyfikacjami oraz miejsca, które te fragmenty łączą pod swoją wyłączną protekcją.

Twoim zadaniem jest doradzić królowi Bajtezjanowi, których lordów wybrać na Obrońców, by zapewnić ochronę dróg Królewskiego Traktu o jak największej łącznej długości

Dane wejściowe

Do zaimplementowanej funkcji przekazane zostaną następujące argumenty:

- *N* - ilość miejsc w Bajtycji
- lista zawierająca krotki postaci *(a, b, s)*, gdzie *a*, *b* to numery miejsc, a *s* to długość łączącego je odcinka drogi
- lista zawierająca dla każdego lorda listę miejsc, w których ma swoje forty. Miejsca są indeksowane od *1*, tj. mają indeksy *1*, *2*, ..., *N*.

Przykładowe wywołanie może wyglądać następująco:

```
solve(6, [
    (1, 2, 4),
    (2, 3, 5),
    (3, 4, 6),
    (4, 5, 8),
    (5, 6, 7),
    (1, 6, 9),
    (2, 5, 10)],
  [
    [1, 3],
    [2, 5],
    [4, 6]])
```

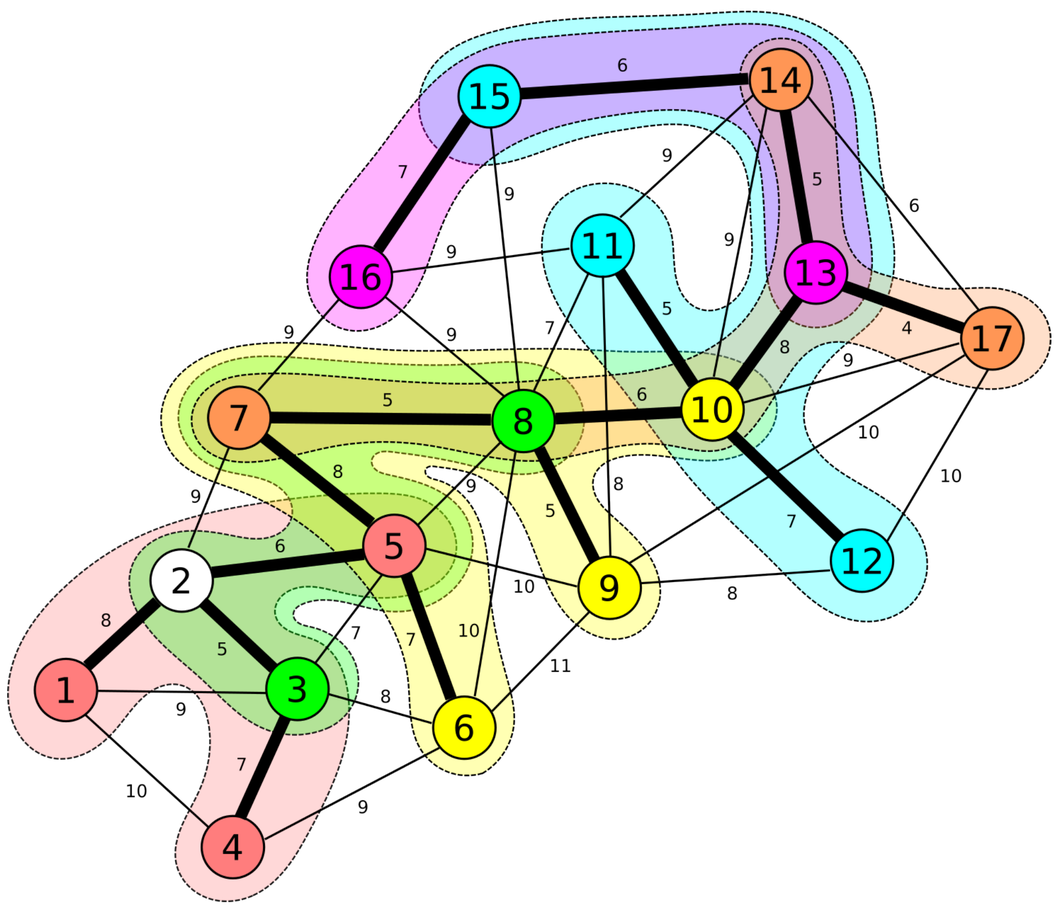
Funkcja *solve* powinna zwracać największą możliwą łączną długość odcinków Królewskiego Traktu, które mogą być chronione przy zachowaniu opisanych wyżej kryteriów.

Dla powyższych danych pierwszy lord z fortami w miejscach *1* i *3* może bronić odcinków drogi *(1, 2)* oraz *(2, 3)* o łącznej długości *9*, drugi - odcinków *(2, 3)*, *(3, 4)* i *(4, 5)* o łącznej długości *19*, a trzeci - odcinków *(4, 5)*, *(5, 6)* o łącznej długości *15*. Jeśli drugi lord zostanie wybrany na obrońcę, nie możemy wybrać pozostałych dwóch, jako że współdzieliliby miejsca i odcinki drogi. Optymalne jest zatem wybranie pierwszego i ostatniego lorda, co pozwala pokryć odcinki *(1, 2)*,, *(2, 3)*, *(4, 5)*, *(5, 6)* o łącznej długości *24*.

Można przyjąć, że graf reprezentujący drogi Bajtycji jest spójny, a każdy lord posiada co najmniej *2* forty.

Przykład

Rozważmy następujący układ miejsc i fortów lordów:



Drogi Króleskiego Traktu są pogrubione. Mamy 6 lordów:

- Lord 1, z fortami w *1*, *4* i *5*, łączna długość dróg: *26*
- Lord 2, z fortami w *3* i *8*, łączna długość dróg: *24*
- Lord 3, z fortami w *6*, *9* i *10*, łączna długość dróg: *31*
- Lord 4, z fortami w *11*, *12* i *15*, łączna długość dróg: *31*
- Lord 5, z fortami w *7*, *14* i *17*, łączna długość dróg: *28*
- Lord 6, z fortami w *13* i *16*, łączna długość dróg: *18*

Lordowie *3* i *4* nie współdzielą żadnego odcinka drogi, ale mają wspólne miejsce *10*, przez które przechodzić musiałyby oddziały obu lordów, zatem nie można wybrać ich obu. Nie ma również możliwości wybrania lorda *5* i jednego z lordów *3* i *4*, co dałoby łączną długość *59*. Mamy jednak możliwość wybrania lordów *1* i *4*, którzy pokrywają łącznie odcinki drogi o długości *57*, i jest to największa łączna długość jaką jesteśmy w stanie pokryć przy takim układzie fortów lordów.

Dla takiej instancji problemu prawidłowa odpowiedź wynosi zatem *57*.

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 3.12.1.
- Program nie może wykorzystywać zewnętrznych bibliotek (biblioteka standardowa jest dopuszczalna)