algograf

Projekt 1: Jarmark

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 2022/2023 - projekty
- hasło: galgo2223

Projekt zostanie oceniony w następujący sposób:

- osoby z najlepszymi programami (15 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: **31 grudnia 2022, 23:59**

Treść zadania

W obliczu narastających napięć społecznych spowodowanych wewnętrznymi problemami krainy, Król Bajtynius postanowił poprawić wizernek Korony organizując dla ludu ogromny jarmark, którego główną atrakcją będą występy cyrkowych sztukmistrzów. Aby odnieść dostateczny efekt, ilość występów podczas jarmarku musi być odpowiednio duża. Tego typu przedsięwzięcie wiąże się rzecz jasna z rozmaitymi wydatkami. Jako jednak że królewski skarbiec świeci ostatnimi czasy pustkami, Bajtynius chciałby osiągnąć swój cel możliwie jak najmniejszym kosztem.

Możliwe do zorganizowania są różnego rodzaju pokazy - każdy ze sztukmistrzów potencjalnie biorących udział w jarmarku może wziąć udział pewnym podzbiorze pokazów, zależnie od swoich umiejętności. Na cenę zorganizowania wystąpienia danego sztukmistrza w danym pokazie składa się koszt ekwipunku (zależny od pokazu), jak również wynagrodzenie sztukmistrza. Wynagrodzenie składa się ze stawki bazowej (zależnej od sztukmistrza), oraz bonusu (zależnego od pokazu i sztukmistrza). Jeden sztukmistrz może zazwyczaj wziąć udział w kilku różnych pokazach, natomiast woli skupić się na jednym - stawka bazowa za udział w każdym następnym pokazie jest wyższa niż w poprzednim.

W ramach powierzonej Ci roli Głównego Koordynatora Wielkiego Jarmarku Bajtycji, na Twoich barkach spoczywa zaplanowanie wydarzenia tak, by osiagnąć wymaganą liczbę występów przy

jak najmniejszym obciążeniu budżetu królestwa.

Dane wejściowe

Do zaimplementowanej funkcji przekazane zostaną następujące argumenty:

- *N* - ilość sztukmistrzów
- *M* - ilość możliwych do zorganizowania pokazów
- *K* - wymagana łączna ilość wystąpień
- lista zawierająca dla każdego sztukmistrza listę wynagrodzeń bazowych za udział kolejno w *1*, *2*, itd. występach, aż do maksymalnej ilości którą skłonny jest wykonać dany artysta
- lista zawierająca dla każdego sztukmistrza listę par zawierających indeks pokazu, w którym może wziąć udział, jak również bonus za dany pokaz
- lista zawierająca dla każdego pokazu cenę przygotowania ekwipunku dla pojedynczego uczestnika Pokazy są indeksowane od *1*, tj. mają indeksy *1*, *2*, …, *M*

Przykładowe wywołanie może wyglądać następująco:

```
solve(3, 4, 5
  [(3, 8), (1, 4, 11), (5, 13)],   # wynagrodzenia bazowe
  [
    [(1, 15), (2, 7)],             # pokazy pierwszego artysty
    [(2, 11), (3, 17)],            # pokazy drugiego artysty
    [(1, 25), (2, 17), (4, 31)]    # pokazy trzeciego artysty
  ],
  [5, 3, 10, 7]                    # ceny ekwipunku dla pokazów
)
```

Takie wywołanie oznacza, że mamy 4 artystów, 3 pokazy, a opłacić chcemy łącznie 5 wystąpień (niektórzy sztukmistrze będą więc musieli wystąpić więcej niż raz). Dla przykładu, jeśli pierwszy sztukmistrz wystąpi wyłącznie w pokazie nr. 1, będzie to kosztować 3 + 15 = 18 (wynagrodzenie bazowe za pojedynczy pokaz + bonus za występ w pokazie nr. 1). Jeśli wystąpi w pokazach 1 i 2, będzie to kosztować 8 + 15 + 7 (wynagrodzenie bazowe za dwa pokazy + bonus za występ w pokazie nr. 1 + bonus za występ w pokazie nr. 2).

Zaimplementowana funkcja powinna zwracać liczbę całkowitą - minimalny łączny koszt zorganizowania wymaganych *K* występów. Dla powyższego wywołania powinna zwrócić 108, wybierając następujące występy:

- artysta 1: występ 1, 2, koszt 8 + (15 + 5) + (7 + 3) = 38
- artysta 2: występ 2, 3, koszt 4 + (11 + 3) + (17 + 10) = 45
- artysta 3: wystep 2: koszt 5 + (17 + 3) = 25

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)

Pliki

- project1.zip