Algorytmy i Struktury Danych
Zadanie offline 4 (19.IV.2022)

Format rozwiązań

Rozwiązanie zadania musi się składać z **krótkiego** opisu algorytmu (wraz z uzasadnieniem poprawności) oraz jego implementacji. Zarówno opis algorytmu jak i implementacja powinny się znajdować w tym samym pliku Pythona (rozszerzenie `.py`). Opis powinien być na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Opis nie musi być długi—wystarczy kilka zdań, jasno opisujących ideę algorytmu. Implementacja musi być zgodna z szablonem kodu źródłowego dostarczonym wraz z zadaniem. Niedopuszczalne jest w szczególności:

1. korzystanie z zaawansowanych struktur danych (np. słowników czy zbiorów),
2. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
3. modyfikowanie testów dostarczonych wraz z szablonem,
4. wypisywanie na ekranie jakichkolwiek napisów innych niż wypisywane przez dostarczony kod (ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania).

Dopuszczalne jest natomiast:

1. korzystanie z następujących elementarnych struktur danych: krotka, lista,
2. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są).
3. korzystanie z wbudowanych funkcji sortujących (można założyć, że mają złożoność $O(n \log n)$).

Wszystkie inne algorytmy lub struktury danych wymagają implementacji przez studenta. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych w pliku z szablonem rozwiązania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 punktów. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Testowanie rozwiązań

Żeby przetestować rozwiązanie zadania należy wykonać polecenie: `python3 zad4.py`

Zadanie offline 4.

Szablon rozwiązania: `zad4.py`

Inwestor planuje wybudować nowe osiedle akademików. Architekci przedstawili projekty budynków, z których inwestor musi wybrać podzbiór spełniając jego oczekiwania. Każdy budynek reprezentowany jest jako prostokąt o pewnej wysokości h, podstawie od punktu a do punktu b, oraz cenie budowy w (gdzie h, a, b i w to liczby naturalne, przy czym $a < b$). W takim budynku może mieszkać $h \cdot (b - a)$ studentów.

Proszę zaimplementować funkcję:

```
def select_buildings(T, p):
    ...
```

która przyjmuje:

- Tablicę `T` zawierająca opisy `n` budynków. Każdy opis to krotka postaci `(h, a, b, w)`, zgodnie z oznaczeniami wprowadzonymi powyżej.
- Liczbę naturalną `p` określającą limit łącznej ceny wybudowania budynków.

Funkcja powinna zwrócić tablicę z numerami budynków (zgodnie z kolejnością w `T`, numerowanych od 0), które nie zachodzą na siebie, kosztują łącznie nie więcej niż `p` i mieszczą maksymalną liczbę studentów. Jeśli więcej niż jeden zbiór budynków spełnia warunki zadania, funkcja może zwrócić dowolny z nich. Dwa budynki nie zachodzą na siebie, jeśli nie mają punktu wspólnego.

Można założyć, że zawsze istnieje rozwiązanie zawierające co najmniej jeden budynek. Funkcja powinna być możliwie jak najszybsza i zużywać jak najmniej pomięci. Należy bardzo skrótowo uzasadnić jej poprawność i oszacować złożoność obliczeniową.

Przykład. Dla argumentów:

```
T = [ (2, 1, 5, 3),
      (3, 7, 9, 2),
      (2, 8, 11, 1) ]
p = 5
```

wynikiem może być tablica: `[ 0, 2 ]`