NAI – mały projekt programistyczny 6

Dyskretny problem plecakowy

Napisz program, rozwiązujący dyskretny problem plecakowy.

Program powinien wczytywać dane ze standardowego wejścia, w następującym formacie:

- w pierwszej linii: jedna liczba całkowita *poj* – pojemność plecaka (w kilogramach);
- w drugiej linii: jedna liczba całkowita *n* – liczba dostępnych przedmiotów do wyboru;
- w kolejnych *n* liniach: dwie liczby całkowite *val weight* oddzielone pojedynczą spacją, oznaczające odpowiednio wartość przedmiotu oraz jego wagę (w kilogramach).

Program powinien:

1. Znaleźć optymalne rozwiązanie problemu plecakowego (tzn. taki wybór przedmiotów, który mieści się w plecaku i ma maksymalną możliwą łączną wartość).
2. Wypisać wybrane przedmioty, po jednym w każdej linii, w formacie takim, jak na wejściu (tzn. *val weight*, oddzielone pojedynczą spacją).

Założenia techniczne:

- Wszystkie liczby na wejściu będą całkowite, dodatnie i mniejsze niż 1000.
- Dodatkowo *n* będzie nie większe niż 25.

Proszę postarać się o rozsądny czas działania (poniżej 5 minut) nawet dla *n* = 25.

Przykład: Dla wejścia

```
30
5
10 5
15 7
53 26
20 11
25 13
```

program powinien wypisać:

```
10 5
20 11
25 13
```

(Kolejność linii może być inna).

Uwaga: Proszę zachowywać specyfikację wejścia i wyjścia z treści zadania, tzn.:

- nie wpisywać na sztywno danych wejściowych do kodu
- rozróżniać argumenty wywołania (np. `moj_program arg1 arg2`) oraz dane pobierane z wejścia (np. `moj_program`, po czym użytkownik wpisuje: `30`, potem `5` itd.) - w tym zadaniu dość wyjątkowo pobieramy dane z wejścia (wyjątek pojawi się w bonusie poniżej)
- wypisywać wynik w formacie opisanym powyżej (a nie zupełnie innym)

To oczywiście nie są kluczowe sprawy, radzę sobie z uruchmianiem Państwa programów tak czy siak, ale myślę, że warto wyrobić sobie nawyk dostarczania produktu działającego zgodnie z zamówieniem :)

Bonus (za dodatkowy punkt z aktywności):

Bonus dotyczy dalszej optymalizacji programu, dla jeszcze wyższych n. W ogólnej sytuacji niewiele możemy tu zdziałać; jak wiadomo z wykładu, najlepsze znane algorytmy działają w czasie wykładniczym. Jednak jeśli założymy dodatkowo, że złodziej się spieszy i nie zdąży zabrać więcej niż k przedmiotów, to okazuje się, że istnieje rozwiązanie w czasie $O(n^k)$. I na tym właśnie polega bonus. Konkretnie:

- Proszę wyposażyć swój program w opcjonalny argument, k, przyjmowany z linii poleceń. (Format wejścia i wyjścia pozostaje taki sam, jak w wersji podstawowej).
- Jeśli k nie zostanie podane, program powinien w dalszym ciągu rozwiązywać wersję podstawową zadania.
- Jeśli k zostanie podane, oznacza ono maksymalną liczbę przedmiotów, które może wziąć złodziej. Program powinien znajdować optymalne rozwiązanie przy tym dodatkowym założeniu. Przy tym proszę postarać się o rozsądny czas działania (poniżej 5 minut) dla $n = 50$ oraz $k = 5$.

Uwaga: Argument k powinien być rzeczywiście argumentem wywołania i być opcjonalny z punktu widzenia użytkownika programu, tzn. program powinien działać, jeśli się go poda lub nie poda (zgodnie z opisem w punktach powyżej).

Uwaga: Program należy wykonać samodzielnie. **Plagiat** lub **niezrozumienie** rozwiązania skutkuje **brakiem zaliczenia projektu**.

Nie można korzystać z gotowych bibliotek do generowania ciągów / zbiorów tudzież rozwiązywania problemu plecakowego. Wszystkie szczegóły algorytmu należy samodzielnie przećwiczyć kodując.

Termin: **5 czerwca**