Bartłomiej Barszczak
WEAIiIB Automatyka i Robotyka
Rok II semestr IV grupa 3

Zadanie 1

```cpp
#include <iostream>
#include <vector>


struct Element {
    int wage;
    int profit;
    int id;
};

/**
 * Funckja wypisuje macierz na konsoli
 * @param matrix macierz
 * @param stage etap
 */
void print_matrix(const std::vector<std::vector<int>> &matrix, int stage) {
    std::cout << "Stage: " << stage << std::endl; // wypisanie etapu

    // wypisanie elementow macierzy
    for (int i = 0; i < matrix[0].size(); i++) {
        for (int j = 1; j < matrix.size(); j++) {
            std::cout << matrix[j][i] << " ";
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}

/**
 * Funckja wypisuje wektor na konsoli
 * @param vector wektor
 * @param title tytul
 */
void print_vector(const std::vector<Element> &vector, const std::string &title) {
    std::cout << title << std::endl; // wyspianie tytulu wektora

    // wyspisanie elementow wektora
    for (auto elem: vector)
        std::cout << "ID: " << elem.id << ", wage: " << elem.wage << ", profit: "
<< elem.profit << std::endl;
//    std::cout << std::endl;
}

/**
 * Fukcja rozwiazujaca problem binarny 0-1 za pomoca programowania dynamicznego
 * @param elements kontener obiektow typu Element
 * @param limit limit (w problemie plecakowym ilosc dostepnego miejsca)
 * @return najwiekszy zysk
 */
int DP_FPLP(std::vector<Element> &elements, int limit) {
```

```
    std::vector<std::vector<int>> result(elements.size() + 1,
                                         std::vector<int>(limit + 1, 0)); //
zainicjalizowanie macierzy wynikowej 0
    std::vector<Element> items = {}; // rzeczy ktore sie wliczaja (w problemie
plecakowych to te ktore nalezy zabrac)
    int stage = (int) elements.size(); // zainicjalizowanie etapu
    int include, column; // zmienne pomocnicze

    //glowna petla funkcji
    for (int i = 1; i <= elements.size(); i++) {
        for (int j = 0; j <= limit; j++) {
            if (elements[i - 1].wage <= j) // sprawdzenie czy jest mozliwe dodanie
elementu
                // jesli tak to jest dodawany ten ktorego zysk jest wiekszy
                result[i][j] = std::max(elements[i - 1].profit + result[i - 1][j -
elements[i - 1].wage],
                                        result[i - 1][j]);
            else
                result[i][j] = result[i - 1][j]; // dodanie w calkowiego zysku w
danym etapie z poprzedniejszego etapu
        }
        print_matrix(result, --stage); // wypisanie macierzy
    }

    include = result[elements.size()][limit]; // przypisanie najwieszkego zysku
    column = limit;

    // petla odpowadajaca za wyszukanie ktore elementy nalezy wliczyc (w problemie
pleckakowym zabrac)
    for (int i = (int) elements.size(); i > 0 && include > 0; i--) {
        if (include == result[i - 1][column]) // sprawdzanie czy elementy w
kolumine sa takie same
            // jesli tak to przechodzimy dalej
            continue;
        else {
            // jesli nie to do listy rzczy ktore nalezy wliczyc jest dodawany jego
identyfikator (w celu latwijeszesz indetyfikacji) oraz aktualizowane sa zmienne
pomocnicze
//            items.push_back(elements[i - 1]);
            items.insert(items.cbegin(), elements[i - 1]);
            include -= elements[i - 1].profit;
            column -= elements[i - 1].wage;
        }
    }
    print_vector(items, "Rozwiazanie: "); // wypisane rzeczy ktore nalezy wliczyc

    return result[elements.size()][limit]; // zwaracanie wyniku (najwiekszy zysk)
}


int main() {

    // dane
    std::vector<Element> stuff = {{1, 6,  1},
                                  {2, 10, 2},
                                  {3, 12, 3},
                                  {4, 9,  4},
                                  {5, 15, 5},
                                  {6, 9,  6},
                                  {7, 21, 7},
                                  {8, 17, 8},
                                  {9, 13, 9},
                                  {4, 35, 10}};
```

```
    int space = 11; // limit (w problemie plecakowym to pojemnosc plecaka)
    int profit = DP_FPLP(stuff, space);
    std::cout << "Zysk: " << profit << std::endl; // wypisanie najwiekszego zysku

    return 0;
}
```

Zadanie 2

```
Stage: 9
0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0

Stage: 8
0 0 0 0 0 0 0 0 0 0
6 6 0 0 0 0 0 0 0 0
6 10 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0
6 16 0 0 0 0 0 0 0 0

Stage: 7
0 0 0 0 0 0 0 0 0 0
6 6 6 0 0 0 0 0 0 0
6 10 10 0 0 0 0 0 0 0
6 16 16 0 0 0 0 0 0 0
6 16 18 0 0 0 0 0 0 0
6 16 22 0 0 0 0 0 0 0
6 16 28 0 0 0 0 0 0 0
6 16 28 0 0 0 0 0 0 0
6 16 28 0 0 0 0 0 0 0
6 16 28 0 0 0 0 0 0 0
6 16 28 0 0 0 0 0 0 0

Stage: 6
0 0 0 0 0 0 0 0 0 0
6 6 6 6 0 0 0 0 0 0
6 10 10 10 0 0 0 0 0 0
6 16 16 16 0 0 0 0 0 0
6 16 18 18 0 0 0 0 0 0
6 16 22 22 0 0 0 0 0 0
6 16 28 28 0 0 0 0 0 0
6 16 28 28 0 0 0 0 0 0
6 16 28 28 0 0 0 0 0 0
6 16 28 31 0 0 0 0 0 0
6 16 28 37 0 0 0 0 0 0
6 16 28 37 0 0 0 0 0 0

Stage: 5
0 0 0 0 0 0 0 0 0 0
6 6 6 6 6 0 0 0 0 0
6 10 10 10 10 0 0 0 0 0
6 16 16 16 16 0 0 0 0 0
6 16 18 18 18 0 0 0 0 0
6 16 22 22 22 0 0 0 0 0
6 16 28 28 28 0 0 0 0 0
6 16 28 28 28 0 0 0 0 0
6 16 28 28 31 0 0 0 0 0
6 16 28 31 33 0 0 0 0 0
6 16 28 37 37 0 0 0 0 0
6 16 28 37 43 0 0 0 0 0

Stage: 4
0 0 0 0 0 0 0 0 0 0
6 6 6 6 6 6 0 0 0 0
6 10 10 10 10 10 0 0 0 0
6 16 16 16 16 16 0 0 0 0
6 16 18 18 18 18 0 0 0 0
6 16 22 22 22 22 0 0 0 0
6 16 28 28 28 28 0 0 0 0
6 16 28 28 28 28 0 0 0 0
6 16 28 28 31 31 0 0 0 0
6 16 28 31 33 33 0 0 0 0
6 16 28 37 37 37 0 0 0 0
6 16 28 37 43 43 0 0 0 0
```

```
Stage: 3
0 0 0 0 0 0 0 0 0 0
6 6 6 6 6 6 6 0 0 0
6 10 10 10 10 10 10 0 0 0
6 16 16 16 16 16 16 0 0 0
6 16 18 18 18 18 18 0 0 0
6 16 22 22 22 22 22 0 0 0
6 16 28 28 28 28 28 0 0 0
6 16 28 28 28 28 28 0 0 0
6 16 28 28 31 31 31 0 0 0
6 16 28 31 33 33 33 0 0 0
6 16 28 37 37 37 37 0 0 0
6 16 28 37 43 43 43 0 0 0

Stage: 2
0 0 0 0 0 0 0 0 0 0
6 6 6 6 6 6 6 6 0 0
6 10 10 10 10 10 10 10 0 0
6 16 16 16 16 16 16 16 0 0
6 16 18 18 18 18 18 18 0 0
6 16 22 22 22 22 22 22 0 0
6 16 28 28 28 28 28 28 0 0
6 16 28 28 28 28 28 28 0 0
6 16 28 28 31 31 31 31 0 0
6 16 28 31 33 33 33 33 0 0
6 16 28 37 37 37 37 37 0 0
6 16 28 37 43 43 43 43 0 0
```

```
Stage: 1
0 0 0 0 0 0 0 0 0 0
6 6 6 6 6 6 6 6 6 0
6 10 10 10 10 10 10 10 10 0
6 16 16 16 16 16 16 16 16 0
6 16 18 18 18 18 18 18 18 0
6 16 22 22 22 22 22 22 22 0
6 16 28 28 28 28 28 28 28 0
6 16 28 28 28 28 28 28 28 0
6 16 28 28 31 31 31 31 31 0
6 16 28 31 33 33 33 33 33 0
6 16 28 37 37 37 37 37 37 0
6 16 28 37 43 43 43 43 43 0

Stage: 0
0 0 0 0 0 0 0 0 0 0
6 6 6 6 6 6 6 6 6 6
6 10 10 10 10 10 10 10 10 10
6 16 16 16 16 16 16 16 16 16
6 16 18 18 18 18 18 18 18 35
6 16 22 22 22 22 22 22 22 41
6 16 28 28 28 28 28 28 28 45
6 16 28 28 28 28 28 28 28 51
6 16 28 28 31 31 31 31 31 53
6 16 28 31 33 33 33 33 33 57
6 16 28 37 37 37 37 37 37 63
6 16 28 37 43 43 43 43 43 63
```

```
Rozwiazanie:
ID: 1, wage: 1, profit: 6
ID: 2, wage: 2, profit: 10
ID: 3, wage: 3, profit: 12
ID: 10, wage: 4, profit: 35
Zysk: 63
```

Zadanie 3

1. Jakie założenia muszą być spełnione dla wag i zysków
   Wagi oraz zyski powinny być nieujemne.

2. Co się stanie, jeśli te założenia nie spełnimy (modyfikacja sposobu rozwiązania zadania)
   Jeżeli profit będzie wynosił zero to taki element nie potrzebie tylko będzie brany pod uwagę, a gdy będzie ujemny to również nie będzie brany pod uwagę. Gorzej sytuacja wygląda, gdy waga jest ujemna, wtedy program się wysypuje.

3. Jaka jest złożoność obliczeniowa algorytmu?
   Złożoność obliczeniowa: O(S * C), gdzie S – ilość „przedmiotów”, a C – limit.