

# Sprawozdanie

## Ćwiczenie 2 - PD dla WiTi

Krzysztof Górski 245079

prowadzący Dr inż. Mariusz Makuchowski

26.03.2021



Politechnika  
Wrocławska

Proponowana ocena: 3.0

projekt realizowany w ramach kursu Sterowanie Procesami Dyskretnymi

grupa piątek 11:15

# 1 Wstęp

Celem zadania było opracowanie i przetestowanie algorytmu programowania dynamicznego WiTi dla zadanych zestawów danych.

## 2 Algorytm

Programowanie dynamiczne opiera się na podziale rozwiązywanego problemu na pod-problemy względem kilku parametrów. W przypadku omawianym przez nas problem rysuje się w następujący sposób:

- Mamy do wykonania  $n$  zadań na pojedynczej maszynie.
- Jeśli dane zadanie jest spóźnione naliczana jest kara.
- Szukamy uszeregowania o minimalnej sumie kara.

Każde zadanie opisane jest trzema parametrami:

- $p_i$  - czas trwania,
- $w_i$  - waga,
- $d_i$  - porządkany termin zakończenia.

Celem algorytmu jest znalezienie optymalnego rozwiązania z uwzględnieniem podproblemów:

$$F(I) = \min_k F\{(I/\{k\}) + K_k(C(I))\}$$

Gdzie

- $I$  - zbiór zadań
- $F(I)$  - kara optymalnego uszeregowania zadań  $I$
- $K_k(t)$  - kara zadania  $k$  zakończonego w czasie  $t$
- $C(I)$  - długość uszeregowania zadań  $I$

### 3 Kod

```
#include <iostream>
#include <fstream>
#include <algorithm>
using namespace std;
// *****
//definicje globalne
// *****
// definicja typu danych
#define DATA_TYPE int
// zmienne i rozmiary
#define SIZE = 10
int LOOP = 11;
int I_size[11];
// sciezka pliku
string PATH = "./witi.data.txt";
string SOLUTION_PATH = "./solution.txt";
// rozpoznawanie zestawu danych
string DATAN[11] = { "data.10:", "data.11:", "data.12:" ,
"data.13:", "data.14:", "data.15:", "data.16:" ,
"data.17:", "data.18:", "data.19:", "data.20:" };
// *****
// KLASY
// *****
template<typename TYPE>
class DATA {
public:
    TYPE t;
    TYPE w;
```

```

    TYPE r;
    int x; // kolejnosc
};
// *****
// FUNKCJE
// *****
void print_data(int I, DATA<DATA_TYPE>** data) {
    cout << "n=" << I_size[I] << endl;
    for (int i = 0; i < I_size[I]; i++)
    {
        cout << data[I][i].x+1 << ")" << data[I][i].t <<
        " " << data[I][i].w << " " << data[I][i].r << " "
        << endl;
    }
}

void read_from_file(int I, DATA<DATA_TYPE> **data,
fstream &file){
    string s;
    while (s != DATAN[I]) file >> s;
    file >> I_size[I];
    data[I] = new DATA<DATA_TYPE>[I_size[I]];
    for (int i = 0; i < I_size[I]; i++)
    {
        data[I][i].x = i;
        file >> data[I][i].t >> data[I][i].w >> data[I][i].r;
    }
}

void algorytm(int I, DATA<DATA_TYPE>** data)

```

```

{
    int bit = 1 << I_size[I];
    int* F = new int[bit];
    F[0] = 0;
    for (int bit_count = 1; bit_count < bit; bit_count++)
    {
        int pomtime = 0;
        for (int i = 0, b = 1; i < I_size[I]; i++, b *= 2)
        {
            if (bit_count & b)
            {
                pomtime += data[I][i].t;
            }
        }
        F[bit_count] = 999999;
        for (int i = 0, b = 1; i < I_size[I]; i++, b *= 2)
        {
            if (bit_count & b)
            {
                F[bit_count] = min(F[bit_count],
                    F[bit_count - b] + data[I][i].w
                    * max(pomtime - data[I][i].r, 0));
            }
        }
    }
    cout << "solution:␣" << F[bit - 1] << endl;
    cout << endl;
    delete[] F;
}

```

```

// *****
//  MAIN
// *****
int main()
{
    DATA<DATA_TYPE>**data = new DATA<DATA_TYPE>*[10];
    fstream file;
    file.open("data.txt");
    for (int i = 0; i < LOOP; i++)
    {
        read_from_file(i, data, file);
        print_data(i, data);
        algorytm(i, data);

    }
    file.close();
    return 0;
}

```

## 4 Wyniki

Algorytm zadziałał zgodnie z założeniem i dla każdego zestawu danych znalazł rozwiązanie optymalne.

```

n = 10
1) 1 2 748
2) 46 5 216
3) 5 7 673
4) 93 4 514
5) 83 1 52
6) 53 7 7
7) 38 1 413
8) 68 6 922
9) 84 5 91
10) 65 4 694
solution: 766

n = 11
1) 1 2 823
2) 46 5 238
3) 5 7 740
4) 93 4 566
5) 83 1 58
6) 53 7 8
7) 38 1 455
8) 68 6 1014
9) 84 5 100
10) 65 4 764
11) 91 7 286
solution: 799

n = 12
1) 1 2 898
2) 46 5 260
3) 5 7 807
4) 93 4 617
5) 83 1 63
6) 53 7 9
7) 38 1 496
8) 68 6 1106
9) 84 5 109
10) 65 4 833
11) 91 7 312
12) 5 7 390
solution: 742

n = 13
1) 1 2 973
2) 46 5 282
3) 5 7 874
4) 93 4 669
5) 83 1 68
6) 53 7 9
7) 38 1 537
8) 68 6 1198
9) 84 5 118
10) 65 4 903
11) 91 7 338
12) 5 7 422
13) 63 7 1276
solution: 688

```

Rysunek 1: wyniki dla zestawów 10-13

```

n = 14
1) 1 2 1048
2) 46 5 303
3) 5 7 942
4) 93 4 720
5) 83 1 74
6) 53 7 10
7) 38 1 579
8) 68 6 1290
9) 84 5 127
10) 65 4 972
11) 91 7 364
12) 5 7 455
13) 63 7 1374
14) 37 3 1362
solution: 497
i
n = 15
1) 1 2 1122
2) 46 5 325
3) 5 7 1009
4) 93 4 771
5) 83 1 79
6) 53 7 11
7) 38 1 620
8) 68 6 1382
9) 84 5 136
10) 65 4 1041
11) 91 7 390
12) 5 7 487
13) 63 7 1472
14) 37 3 1460
15) 72 7 968
solution: 440

```

```

n = 16
1) 1 2 1197
2) 46 5 347
3) 5 7 1076
4) 93 4 823
5) 83 1 84
6) 53 7 12
7) 38 1 661
8) 68 6 1474
9) 84 5 145
10) 65 4 1111
11) 91 7 415
12) 5 7 520
13) 63 7 1570
14) 37 3 1557
15) 72 7 1032
16) 8 6 1402
solution: 423
i
n = 17
1) 1 2 1272
2) 46 5 368
3) 5 7 1143
4) 93 4 874
5) 83 1 90
6) 53 7 12
7) 38 1 703
8) 68 6 1566
9) 84 5 154
10) 65 4 1180
11) 91 7 441
12) 5 7 552
13) 63 7 1668
14) 37 3 1654
15) 72 7 1097
16) 8 6 1489
17) 27 4 1290
solution: 417

```

Rysunek 2: wyniki dla zestawów 14-18



```

n = 18
1) 1 2 1347
2) 46 5 390
3) 5 7 1211
4) 93 4 926
5) 83 1 95
6) 53 7 13
7) 38 1 744
8) 68 6 1658
9) 84 5 163
10) 65 4 1250
11) 91 7 467
12) 5 7 585
13) 63 7 1767
14) 37 3 1751
15) 72 7 1161
16) 8 6 1577
17) 27 4 1366
18) 48 3 490
solution: 405

n = 19
1) 1 2 1422
2) 46 5 412
3) 5 7 1278
4) 93 4 977
5) 83 1 100
6) 53 7 14
7) 38 1 785
8) 68 6 1751
9) 84 5 173
10) 65 4 1319
11) 91 7 493
12) 5 7 617
13) 63 7 1865
14) 37 3 1849
15) 72 7 1226
16) 8 6 1665
17) 27 4 1442
18) 48 3 517
19) 36 2 915
solution: 393

n = 20
1) 1 2 1496
2) 46 5 433
3) 5 7 1345
4) 93 4 1028
5) 83 1 105
6) 53 7 15
7) 38 1 827
8) 68 6 1843
9) 84 5 182
10) 65 4 1389
11) 91 7 519
12) 5 7 650
13) 63 7 1963
14) 37 3 1946
15) 72 7 1290
16) 8 6 1752
17) 27 4 1518
18) 48 3 544
19) 36 2 963
20) 89 9 119
solution: 897

```

Rysunek 3: wyniki dla zestawów 18-20

## 5 Wnioski

Algorytmy programowania dynamicznego pomimo skomplikowanych założeń dają się dość prosto zaimplementować w iteracyjnej formie. Większy problem stanowi zaimplementowanie permutacji, których nie udało się zastosować w tym rozwiązaniu.

Rozwiązanie programu na ocenę: 3.0

## Spis rysunków

1	wyniki dla zestawów 10-13 . . . . .	7
2	wyniki dla zestawów 14-18 . . . . .	8
3	wyniki dla zestawów 18-20 . . . . .	9