

Optimizacijske metode (IŠRM), 1. domača naloga

Leon Fišer

April 15, 2025

1 Tovarna elektornskih naprav

a) Odločite se, kaj bi bile primerne spremenljivke in za vsako napišite, kaj pomeni.

$$\begin{aligned} D_i &:= \text{Število delavcev, ki bodo sestavljali sprejemnike } i\text{-ti teden.} & (1 \leq i \leq 4) \\ M_{i,j} &:= \text{Število delavcev, ki bodo } i\text{-ti teden učili } j \text{ študentov.} & (1 \leq i, j \leq 3) \\ M_{4,j} &:= 0 & (1 \leq j \leq 3) \\ M_i &:= M_{i,1} + M_{i,2} + M_{i,3} & (1 \leq i \leq 4) \\ C_i &:= \text{Cena enega radija } i\text{-ti teden.} & (1 \leq i \leq 4) \\ P_i &:= \text{Dobiček tovarne } i\text{-ti teden.} & (1 \leq i \leq 4) \end{aligned}$$

b) Zapišite kriterijsko funkcijo in omejitve.

$$\begin{aligned} P_i &= D_i \cdot (50 \cdot C_i - 450\text{€}) - M_{i,1} \cdot 300\text{€} - M_{i,2} \cdot 400\text{€} - M_{i,3} \cdot 500\text{€} & (1 \leq i \leq 4) \\ C_i &= 20\text{€} - (i - 1) \cdot 2\text{€} & (1 \leq i \leq 4) \end{aligned}$$

$$\max \quad P_1 + P_2 + P_3 + P_4$$

p.p.

$$\begin{aligned} D_1 + D_2 + D_3 + D_4 &= 400 \\ D_1 + M_1 &\leq 40 \\ D_2 + M_2 &\leq D_1 + 2 \cdot M_{1,1} + 3 \cdot M_{1,2} + 4 \cdot M_{1,3} \\ D_3 + M_3 &\leq D_2 + 2 \cdot M_{2,1} + 3 \cdot M_{2,2} + 4 \cdot M_{2,3} \\ D_4 + M_4 &\leq D_3 + 2 \cdot M_{3,1} + 3 \cdot M_{3,2} + 4 \cdot M_{3,3} \\ 0 &\leq D_i & (1 \leq i \leq 4) \\ 0 &\leq M_{i,j} & (1 \leq i, j \leq 3) \end{aligned}$$

c) Linearni program rešite s pomočjo računalnika (c++).

Odsek kode c++ implementacije dvofazne simpleksne metode najdete v prilogi.

VHOD

```
13 6
550 -300 -400 -500 450 -300 -400 -500 350 -300 -400 -500 250
1 0 0 0 1 0 0 0 1 0 0 0 1
-1 0 0 0 -1 0 0 0 -1 0 0 0 -1
1 1 1 1 0 0 0 0 0 0 0 0 0
-1 -2 -3 -4 1 1 1 1 0 0 0 0 0
0 0 0 0 -1 -2 -3 -4 1 1 1 1 0
0 0 0 0 0 0 0 0 -1 -2 -3 -4 1
400 -400 40 0 0 0
```

IZHOD

```
128000
0 0 0 40 160 0 0 0 160 0 0 0 80
```

d) Komentirajte rešitev.

Da izdelamo natanko 20000 radijskih sprejemnikov s čim večjim dobičkom, bo vsak delavec v 1-vem tednu izučil 3 študente. V naslednjih 3-eh tednih pa bodo vsi samo delali radijske sprejemnike (razen zadnji teden, ki jih bo pa delalo samo 80, ostale delavce bomo pa poslali na kolektivni dopust in jih ne plačamo). Ta rešitev je optimalna. Dobiček bo 128000€.

2 Družabna igra

a) Zapišite plačilno matriko za gornjo matrično igro. Pri tem si pomagajte z računalnikom (c++).

Naj bo (i, i_2, i_3, i_4) strategija igralca. Igralec bo povedal nasprotniku, da je izbral i -to karto, potem pa, če je nasprotnik izbral j -to karto, bo igralec svojo izbiro popravil na $(i + i_j)$ -to karto. Tedaj naj bo $A \in \mathbb{Z}^{81 \times 81}$ plačilna matrika.

$$\begin{aligned} \Phi(i) &:= ((\lfloor \frac{i}{27} \rfloor, \lfloor \frac{i}{9} \rfloor, \lfloor \frac{i}{3} \rfloor, \lfloor \frac{i}{1} \rfloor) \bmod 3) + (2, -1, -1, -1) \quad (0 \leq i < 81) \\ A_{i,j} &:= B_{\Phi(i)_1 + \Phi(i)_{\Phi(j)_1}, \Phi(j)_1 + \Phi(j)_{\Phi(i)_1}} \quad (0 \leq i, j < 81) \end{aligned}$$

b) Poiščite optimalni strategiji za oba igralca in s pomočjo tega napišite preprosta navodila za igranje igre. Ali je igra poštena?

Problem matrične igre prevedem na linearni program. Ta rešim s pomočjo programa uporabljen v 1. nalogi. Vrednost igre je $-\frac{5}{13} \neq 0$, zato igra *ni* poštena.

1. Igralec:

$$\begin{aligned} \max \quad & s - s' \\ \text{p.p.} \quad & \\ & 1 \cdot (s - s') \leq A^T \cdot x \\ & 1 \cdot x = 1 \\ & x, s, s' \geq 0 \end{aligned}$$

Optimalna strategija:

$$\begin{aligned} P(\Phi(1)) &= \frac{2}{13} \\ P(\Phi(2)) &= \frac{3}{13} \\ P(\Phi(47)) &= \frac{2}{13} \\ P(\Phi(51)) &= \frac{1}{13} \\ P(\Phi(53)) &= \frac{2}{13} \\ P(\Phi(76)) &= \frac{2}{13} \\ P(\Phi(79)) &= \frac{1}{13} \end{aligned}$$

2. Igralec:

$$\begin{aligned} \min \quad & t - t' \\ \text{p.p.} \quad & \\ & 1 \cdot (t - t') \geq A \cdot y \\ & 1 \cdot y = 1 \\ & y, t, t' \geq 0 \end{aligned}$$

Optimalna strategija:

$$\begin{aligned} P(\Phi(4)) &= \frac{2}{39} \\ P(\Phi(19)) &= \frac{2}{39} \\ P(\Phi(36)) &= \frac{7}{26} \\ P(\Phi(38)) &= \frac{5}{26} \\ P(\Phi(39)) &= \frac{3}{26} \\ P(\Phi(58)) &= \frac{5}{39} \\ P(\Phi(73)) &= \frac{5}{78} \\ P(\Phi(76)) &= \frac{5}{39} \end{aligned}$$

IMPLEMENTACIJA DVOFAZNE SIMPLEKSNE METODE V C++

```

/* *****
 *      author: omega
 *      created: 03-02-2025
 *      ***** */

#include <bits/stdc++.h>

using namespace std;

struct buffer : streambuf {
    int overflow(int x) {
        return x;
    }
};

static buffer null_buffer;
static ostream null_stream(&null_buffer);

void local() {
    #if defined(LOCAL_FLAG)
        freopen("/home/omega/Documents/input.in", "r", stdin);
        freopen("/home/omega/Documents/output.out", "w", stdout);
        freopen("/home/omega/Documents/error.err", "w", stderr);
        #define debug cerr
    #else
        #define debug null_stream
    #endif
}

#define ff first
#define ss second
#define pb emplace_back
#define pob pop_back
#define pf emplace_front
#define pof pop_front
#define endl '\n'

typedef long long ll;
typedef unsigned long long ull;
typedef unsigned int uint;
typedef pair<int, int> ii;
typedef tuple<int, int, int> iii;
typedef pair<ll, ll> pll;
typedef tuple<ll, ll, ll> tlll;
typedef complex<double> cd;
typedef long double ld;

const ll mod = 11(1e9 + 7);
const int maxn = int(1e6) + 10;
const int maxm = int(1e3) + 10;
const int lg = 31;
const long double eps = (long double)(1e-9);

struct ulomek {
    ll num, den;

    ulomek(): num(0), den(1) {}

    ulomek(ll num, ll den) : num(num), den(den) {
        normalize();
    }

    void normalize() {
        if(num == 0) den = 1;
        if(den < 0) {
            den *= -1;
            num *= -1;
        }
        ll d = gcd(num, den);
        num /= d;
        den /= d;
    }

    ulomek operator-() const {
        return ulomek(-num, den);
    }

    ulomek& operator+=(const ulomek &drugi) {
        num = num * drugi.den + drugi.num * den;
        den = den * drugi.den;
        normalize();
        return *this;
    }

    ulomek& operator-=(const ulomek &drugi) {
        num = num * drugi.den - drugi.num * den;
        den = den * drugi.den;
        normalize();
        return *this;
    }

    ulomek& operator*=(const ulomek &drugi) {
        num = (num * drugi.num);
        den = (den * drugi.den);
        normalize();
        return *this;
    }

    ulomek& operator/=(const ulomek &drugi) {
        num = (num * drugi.den);
        den = (den * drugi.num);
        normalize();
        return *this;
    }
};

ulomek operator+(ulomek prvi, const ulomek &drugi) {
    prvi += drugi;
    return prvi;
}

ulomek operator-(ulomek prvi, const ulomek &drugi) {
    prvi -= drugi;

```

```

    return prvi;
}
ulomek operator*(ulomek prvi, const ulomek &drugi) {
    prvi *= drugi;
    return prvi;
}
ulomek operator/(ulomek prvi, const ulomek &drugi) {
    prvi /= drugi;
    return prvi;
}

bool operator==(const ulomek &prvi, const ulomek &drugi) {
    return (prvi.num == drugi.num) && (prvi.den == drugi.den);
}

bool operator<(const ulomek &prvi, const ulomek &drugi) {
    return (prvi.num * drugi.den) < (prvi.den * drugi.num);
}

bool operator<=(const ulomek &prvi, const ulomek &drugi) {
    return (prvi < drugi) || (prvi == drugi);
}

bool operator>(const ulomek &prvi, const ulomek &drugi) {
    return drugi < prvi;
}

bool operator>=(const ulomek &prvi, const ulomek &drugi) {
    return drugi <= prvi;
}

ostream& operator<<(ostream &out, const ulomek &prvi) {
    if(prvi.den == 1) {
        out << prvi.num;
    } else {
        out << prvi.num << "/" << prvi.den;
    }
    return out;
}

istream& operator>>(istream &in, ulomek &prvi) {
    in >> prvi.num; prvi.den = 1;
    return in;
}

ulomek a[maxm][maxm], b[maxm], c[maxm];

int base[maxm];

ulomek x[maxm][maxm];

void vstavi(int i, int j) {
    for(int k = 0; k < maxm; k++) {
        x[j][k] += x[j][i] * x[i][k];
    }
    x[j][i] = ulomek();
}

void izpostavi(int i, int j) {
    for(int k = 0; k < maxm; k++) {
        x[j][k] = x[i][k] / (-x[i][j]);
    }
    x[j][j] = ulomek();
    x[j][i] = (ulomek(1,1) / x[i][j]);
}

void print(int n) {
    cout << x[0][0] << "\n";

    for(int i = 1; i <= n; i++) {
        cout << (base[i] ? x[i][0] : ulomek()) << " ";
    }

    cout << "\n";
}

void simplex(int n, int m, int faza) {
    int w = n+m+1;

    if(faza == 1) {
        for(int i = 1; i <= m; i++) {
            x[n+i][0] = b[i]; base[n+i] = 1;
            for(int j = 1; j <= n; j++) {
                x[n+i][j] = -a[i][j];
            }
        }

        x[0][w] = ulomek(-1,1);

        for(int i = 1; i <= m; i++) {
            x[n+i][w] = ulomek(1,1);
        }

        int pivot = 0;

        for(int i = n+1; i <= n+m; i++) {
            if(!pivot && (x[i][0] < ulomek())) {
                pivot = i;
            }
            if(pivot && (x[i][0] < x[pivot][0])) {
                pivot = i;
            }
        }

        if(!pivot) return;

        izpostavi(pivot, w);
        base[w] = 1; base[pivot] = 0;
    }
}

```

```

        for(int i = 0; i <= n+m+1; i++) {
            vstavi(w, i);
        }
    }

    if(faza == 2) {
        if(base[w]) {
            cout << "nedopusten\n"; return;
        }

        for(int i = 1; i <= n+m; i++) {
            if(base[i] && x[i][0] < ulomek()) {
                cout << "nedopusten\n"; return;
            }
        }

        for(int j = 1; j <= n; j++) {
            x[0][j] = c[j];
        }

        for(int j = 1; j <= m; j++) {
            x[0][n+j] = ulomek();
        }

        for(int i = 0; i <= n+m; i++) {
            x[i][w] = ulomek();
        }

        base[w] = 0;

        for(int i = 0; i <= n; i++) {
            if(base[i]) vstavi(i, 0);
        }
    }

    int tr = 1000;

    while(tr--) {

        int vhd = 0;

        for(int j = 1; j <= n+m; j++) {
            if(x[0][j] > ulomek()) {
                vhd = j; break;
            }
        }

        int pivot = 0;

        if(!vhd) {
            if(faza == 2) print(n);
            return;
        }

        for(int i = 1; i <= n+m+1; i++) {
            if(base[i] && x[i][vhd] < ulomek()) {
                if(!pivot || (x[pivot][0] / x[pivot][vhd] <= x[i][0] / x[i][vhd])) {
                    pivot = i;
                }
            }
        }

        if(!pivot) {
            cout << "neomejen\n"; return;
        }

        izpostavi(pivot, vhd);
        base[vhd] = 1; base[pivot] = 0;

        for(int i = 0; i <= n+m+1; i++) {
            vstavi(vhd, i);
        }

    }

    debug << "error\n";
}

void solve() {

    int n, m; cin >> n >> m;

    for(int i = 1; i <= n; i++) {
        cin >> c[i];
    }

    for(int i = 1; i <= m; i++) {
        for(int j = 1; j <= n; j++) {
            cin >> a[i][j];
        }
    }

    for(int i = 1; i <= m; i++) {
        cin >> b[i];
    }

    simplex(n,m,1); // print(0,0,n,m);
    simplex(n,m,2); // print(0,0,n,m);
}

signed main() {
    local();
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);

    int t = 1; // cin >> t;
    while (t--) {
        solve();
    }
}

```

```
}  
    return 0;  
}
```

Kodo programa sem objavil tudi na spletu preko povezave:

<https://github.com/Enmendurana/Optimizacijske-metode-I-RM-1.-doma-a-naloga/blob/main/dvo%20fazna%20simpleksna%20metoda.cpp>

VHOD (1. Igralec)

```
18 13  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 -1  
0 0 0 0 0 0 3 3 3 3 1 1 1 1 0 0 1 -1  
0 0 3 3 3 3 0 0 0 0 -1 -1 -1 -1 0 0 1 -1  
0 0 2 2 2 2 0 0 0 0 1 1 1 1 0 0 1 -1  
0 0 0 0 1 1 0 0 1 1 0 0 1 1 1 0 1 -1  
0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 2 1 -1  
0 0 3 3 -1 -1 3 3 -1 -1 3 3 -1 -1 1 0 1 -1  
0 0 3 3 -1 -1 3 3 -1 -1 3 3 -1 -1 0 2 1 -1  
3 0 3 -1 3 -1 3 -1 3 -1 3 -1 3 -1 0 0 1 -1  
3 0 -1 0 -1 0 -1 0 -1 0 -1 0 -1 0 0 0 1 -1  
0 2 3 -1 3 -1 3 -1 3 -1 3 -1 3 -1 0 0 1 -1  
0 2 -1 0 -1 0 -1 0 -1 0 -1 0 -1 0 0 0 1 -1  
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0  
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 1 -1
```

IZHOD (1. Igralec)

```
-5/13  
2/13 3/13 0 0 0 0 0 0 0 0 2/13 1/13 2/13 2/13 1/13 0 5/13
```

VHOD (2. Igralec)

```
13 18  
0 0 0 0 0 0 0 0 0 0 0 -1 1  
0 0 0 0 0 0 0 -3 -3 0 0 -1 1  
0 0 0 0 0 0 0 0 0 -2 -2 -1 1  
0 -3 -2 0 0 -3 -3 -3 1 -3 1 -1 1  
0 -3 -2 0 0 -3 -3 1 0 1 0 -1 1  
0 -3 -2 -1 -1 1 1 -3 1 -3 1 -1 1  
0 -3 -2 -1 -1 1 1 0 1 0 -1 1  
-3 0 0 0 -3 -3 -3 1 -3 1 -1 1  
-3 0 0 0 -3 -3 1 0 1 0 -1 1  
-3 0 0 -1 -1 1 1 -3 1 -3 1 -1 1  
-3 0 0 -1 -1 1 1 0 1 0 -1 1  
-1 1 -1 0 0 -3 -3 -3 1 -3 1 -1 1  
-1 1 -1 0 0 -3 -3 1 0 1 0 -1 1  
-1 1 -1 -1 -1 1 1 -3 1 -3 1 -1 1  
-1 1 -1 -1 -1 1 1 0 1 0 -1 1  
0 0 0 -1 0 -1 0 0 0 0 0 -1 1  
0 0 0 0 -2 0 -2 0 0 0 0 -1 1  
1 1 1 1 1 1 1 1 1 0 0  
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 1 -1
```

IZHOD (2. Igralec)

```
5/13  
2/39 0 2/39 7/26 5/26 3/26 0 0 5/39 5/78 5/39 0 5/13
```