問題解決

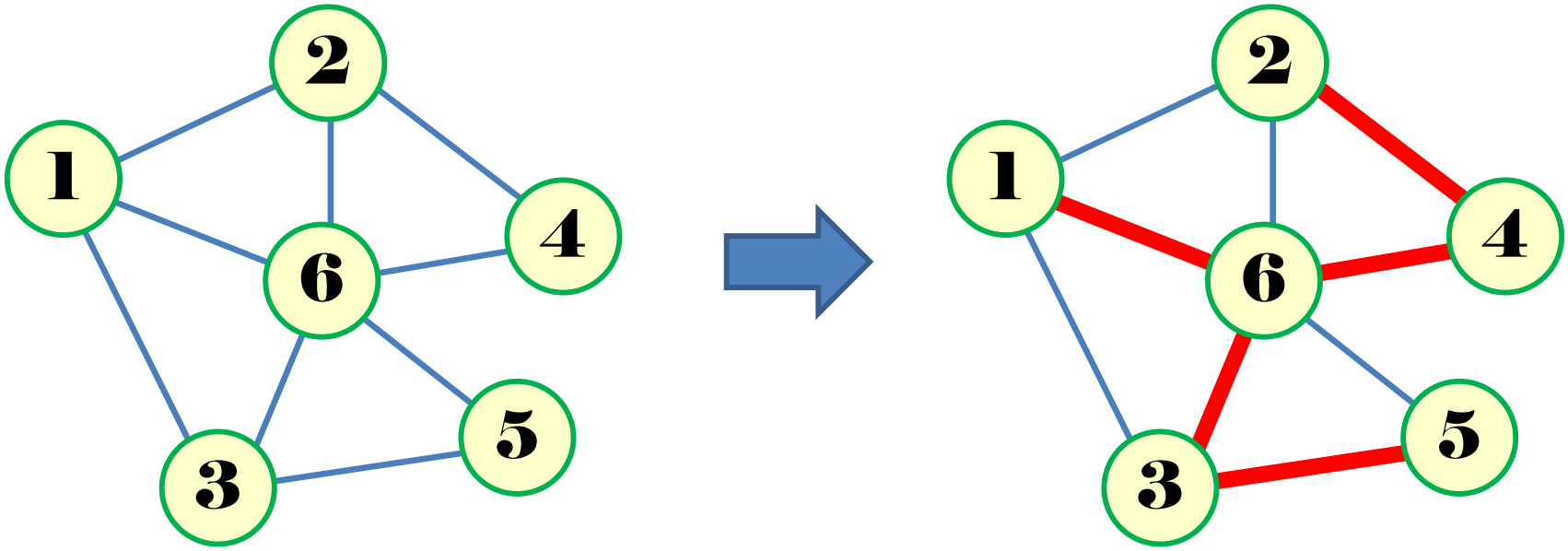
グラフ最適化と整数計画法
5. 最小全域木問題

堀田 敬介

最小全域木問題の最適化

- 最小全域木問題 minimum spanning tree problem
  - 無向グラフ $G = (V, E)$, 点集合 $V = \{1, 2, \ldots, n\}$, 枝集合 E, $|V|=n$, $|E|=m$
  - 各枝 (i, j) にはコスト c_{ij} がある
  - 全域木= 全点に接続する木（全点を張る木）

  - 目的 = コストの総和最小の全域木を求める



所与のグラフ $G=(V, E)$

全域木 spanning tree の例

$T = \{(1,6),(2,4),(3,5),(3,6),(4,6)\}$

最小全域木問題の最適化

無向グラフ $G=(V, E)$ を有向グラフ $G=(V, E)$ に変更する枝(i, j) は両向き．ただし，点1のみ出枝$(1, j)$ だけとする

- 最適化問題の定式化（変数設定）
  - 0-1変数 $x_{ij} = \begin{cases} 1 & \cdots \text{枝}(i,j)\text{を全域木として使う} \\ 0 & \cdots \text{枝}(i,j)\text{を全域木として使わない} \end{cases}$
  - 整数変数 $u_i \in \{0,1,\ldots,n-1\}$ …各点 i のポテンシャル（0以上，n-1以下）

※点集合 $V=\{1, 2, \ldots, n\}$

- 最適化問題の定式化（Σ表記）

$$\min. \sum_{(i,j)\in E} c_{ij} x_{ij}$$

$$\text{s. t. } \sum_{i\in V} x_{ij} = 1 \quad (\forall i \in V - \{1\})$$

点1以外の各点（2～n）について，入枝は1本のみ全域木の枝として採用

$$u_i + 1 \leq u_j + (n-1)(1-x_{ij}) \quad (\forall (i,j) \in E)$$

$$u_1 = 0, 1 \leq u_i \leq n-1 \quad (\forall i \in \{2,\ldots,n\})$$

ポテンシャル制約（閉路禁止制約）

$$x_{ij} \in \{0,1\} \quad (\forall (i,j) \in E)$$

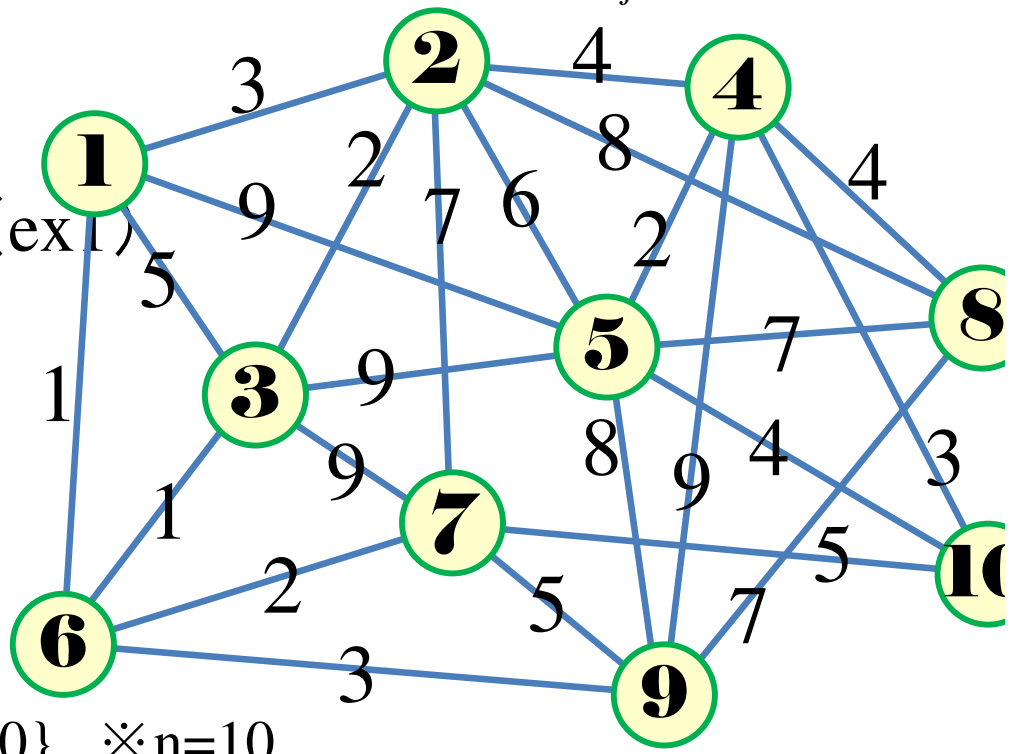
＜ポテンシャル制約について＞

※x_{ij}=0 のとき u_i, u_j は任意．なぜなら，$u_i \leqq u_j + n\text{-}2$ となり，u_i, $u_j \in [1, n\text{-}1]$ より，全てのu_i, u_j で成り立つから

※x_{ij}=1 のとき $u_i + 1 \leqq u_j$ 即ち，枝(i, j)採用時，点 j のポテンシャルは点 i のポテンシャル+1以上．各点のポテンシャルは点1をrootとした木の高さ（rootからの距離）を示す（rootから単純パスとなる木でなければ，uには自由度があるので絶対ではない）

最小全域木問題の最適化

- 最小全域木問題 minimum spanning tree problem
  - 無向グラフ $G = (V, E)$, 点集合 V, 枝集合 E, 枝 (i, j) のコストc_{ij}
  - コスト総和最小の全域木を求める



- 最小全域木でモデル化出来る例(ex1)
  - 10の通信中継点がある
  - 中継点間に通信ケーブルを通す
  - 各ケーブルに利用コストがある
  - 全中継点をつなぐ
  - 中継点集合 = 点集合 $V = \{1, 2, \ldots, 10\}$ ※n=10
  - ケーブル集合 = 枝集合 E = {(1,2),(1,3),(1,5),(1,6),(2,3),(2,4),(2,5),(2,7),(2,8),(3,5),(3,6),(3,7),(4,5),(4,8),(4,9),(4,10),(5,8),(5,9),(5,10),(6,7),(6,9),(7,9),(7,10),(8,9)}
  - 各枝のコスト(3, 5, 9, 1, 2, 4, 6, 7, 8, 9, 1, 9, 2, 4, 9, 3, 7, 8, 4, 2, 3, 5, 5, 7)
  - 目的:最小全域木を求める

最小全域木問題をCPLEXで解く

- 新規プロジェクトの作成
  - ① [ファイル(F)]ー[新規(N)]ー[OPLプロジェクト]を選択
  - ② [プロジェクト名] を記入（例：MinSpannigTree）し，3カ所にチェックする
    - ☑ デフォルトの実行構成の追加
    - ☑ モデルの作成
    - ☑ データの作成
  - ③ [終了]をクリック

プロジェクト名は自由だが，半角英数で何の問題を解こうとしているのかが分かる名前が良い

- プロジェクト内のいくつかの名前を変更
  - ✓ [構成1] → [config1]　※日本語を英語に変更しないと実行時エラーになる
  - ✓ モデルファイル [MinSpanningTree.mod] → [mst.mod]
  - ✓ データファイル [MinSpanningTree.dat] → [mstex1.dat]
- 空のExcelファイル[mst.xlsx]を作り，プロジェクト内にドラッグ&ドロップする（※これでプロジェクトの保存フォルダにコピーされる）
- モデルファイル・データファイルを記述し保存（次ページ参照）
- [config1]にモデルファイルとデータファイルをセットし，解く

最小全域木問題をCPLEXで解く

- モデルファイル（mst.mod）の中身の記述

1つのファイル「mst.mod」に
①②の順に記述して保存

①

```
int v_max = ...;// 点数|V|
int e_max = ...;// 枝数|E|

range V = 1..v_max;
range E = 1..e_max;

tuple Edge {
  int i;
  int j;
  int cost;
}
Edge edge[E] = ...;

dvar int+ x[E] in 0..1;
dvar int+ u[V] in 0..v_max-1;
```

②

```
minimize
  sum(e in E) edge[e].cost * x[e];
subject to {
  forall (v in 2..v_max)
    sum(e in E:edge[e].j==v) x[e] == 1;
  } // 各点2～n への全域木で採用する入枝数は1本

  u[1] == 0;     // 点1のポテンシャルは0
  forall (v in 2..v_max) {
    u[v] >= 1;   // 点2～nのポテンシャルは1以上（v_max-1以下）
  }
  forall (e in E) {
    u[edge[e].i] + 1 <= u[edge[e].j] + (v_max-1)*(1-x[e]);
  }
}
```

最小全域木問題をCPLEXで解く

- データファイル(mstex1.dat)の中身の記述

```
v_max = 10;// 点数|V|
e_max = 44;// 枝数|E|

edge = [// <i,j,cost>
<1,2,3>,<1,3,5>,<1,5,9>,<1,6,1>,
<2,3,2>,<3,2,2>,<2,4,4>,<4,2,4>,<2,5,6>,<5,2,6>,<2,7,7>,<7,2,7>,<2,8
,8>,<8,2,8>,
<3,5,9>,<5,3,9>,<3,6,1>,<6,3,1>,<3,7,9>,<7,3,9>,
<4,5,2>,<5,4,2>,<4,8,4>,<8,4,4>,<4,9,9>,<9,4,9>,<4,10,3>,<10,4,3>,
<5,8,7>,<8,5,7>,<5,9,8>,<9,5,8>,<5,10,4>,<10,5,4>,
<6,7,2>,<7,6,2>,<6,9,3>,<9,6,3>,
<7,9,5>,<9,7,5>,<7,10,5>,<10,7,5>,
<8,9,7>,<9,8,7>
];

SheetConnection sheet("mst.xlsx");
edge to SheetWrite(sheet, "Sheet1!A2:C45");
x to SheetWrite(sheet, "Sheet1!D2:D45");
u to SheetWrite(sheet, "Sheet1!G2:G11");
```

点1のみ出枝$(1, j)$ だけ
(1,2), (1,3), (1,5), (1,6)

3つ目の数値は cost

点2～nの枝(i, j) は両向き
(2,3), (3,2), (2,4), (4,2),
(2,5), (5,2), (2,7), (7,2),
…

計算結果を
Excelファイル[mst.xlsx]
に出力

最小全域木問題をCPLEXで解く

- 結果（[解]タブ）

最小コストは22（= 1+2+4+1+2+4+3+2+3）

```
// solution (optimal) with objective 22
// Quality Incumbent solution:
// MILP objective                                2.2000000000e+01
// MILP solution norm |x| (Total, Max)           3.80000e+01  5.00000e+00
// MILP solution error (Ax=b) (Total, Max)       0.00000e+00  0.00000e+00
// MILP x bound error (Total, Max)               0.00000e+00  0.00000e+00
// MILP x integrality error (Total, Max)         0.00000e+00  0.00000e+00
// MILP slack bound error (Total, Max)           0.00000e+00  0.00000e+00

x = [0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0];
u = [0 3 2 4 5 1 2 5 2 5];
```

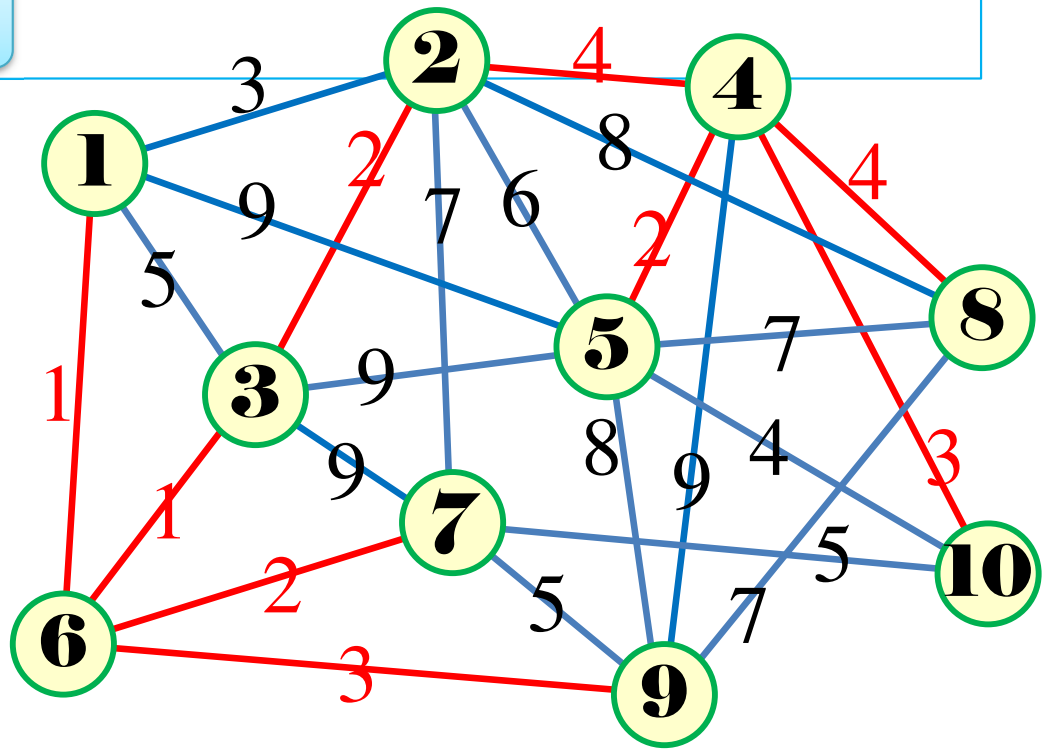
最適値 = 22

最適解

- 枝集合E={
  (1,2),(1,3),(1,5),(1,6),(2,3),
  (2,4),(2,5),(2,7),(2,8),(3,5),
  (3,6),(3,7),(4,5),(4,8),(4,9),
  (4,10),(5,8),(5,9),(5,10),(6,7),
  (6,9),(7,9),(7,10),(8,9)}

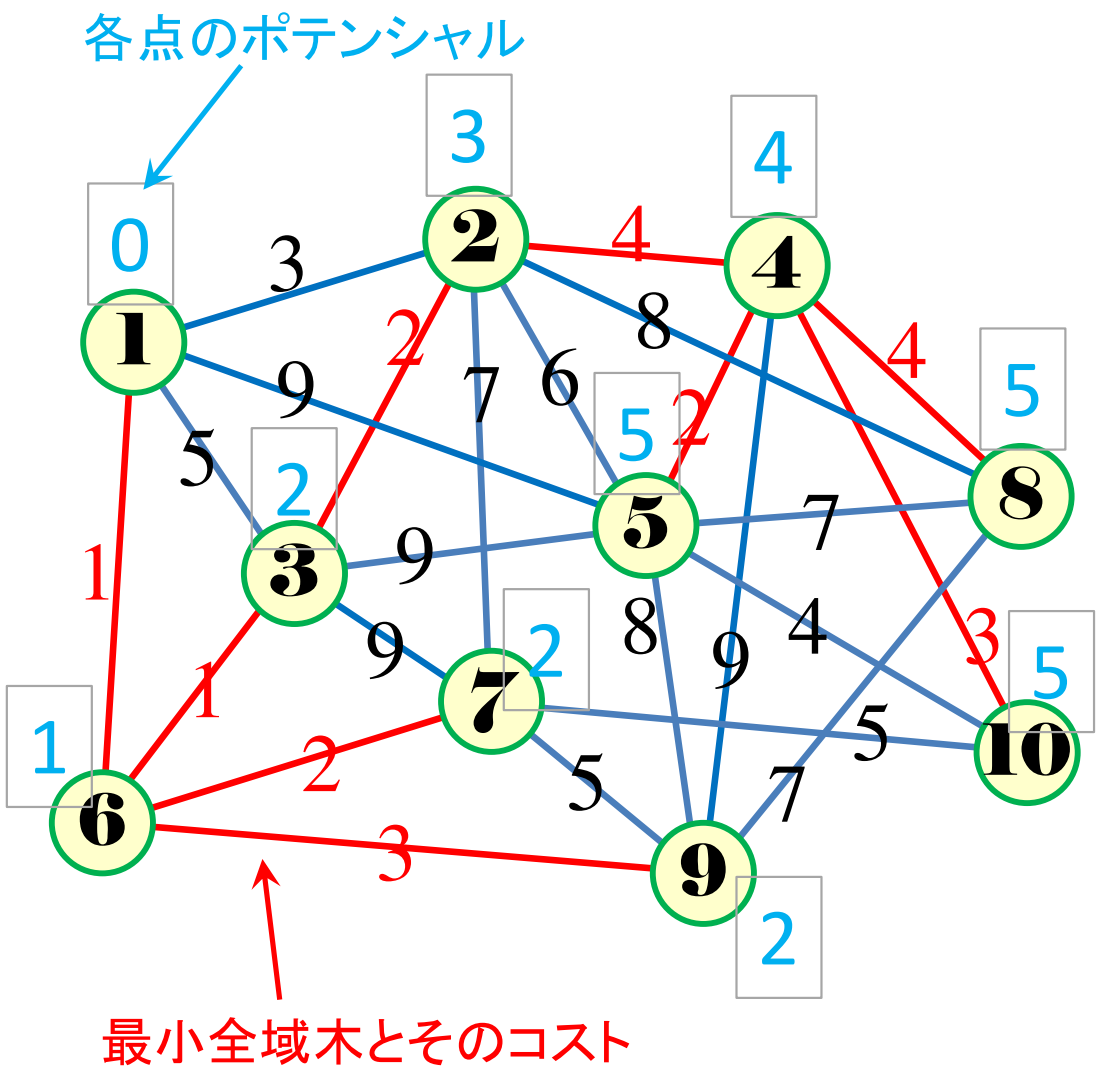
全域木の枝集合T（＝赤色）

使用枝数は 9本（=n-1）



最小全域木問題をCPLEXで解く

- 結果（Excelファイル[mst.xlsx]）



	A	B	C	D	E	F	G	H
1	(i,	j)	cost	x[i,j]		v	u[v]	
2	1	2	3	0		1	0	
3	1	3	5	0		2	3	
4	1	5	9	0		3	2	
5	1	6	1	1		4	4	
6	2	3	2	0		5	5	
7	3	2	2	1		6	1	
8	2	4	4	1		7	2	
9	4	2	4	0		8	5	
10	2	5	6	0		9	2	
11	5	2	6	0		10	5	
12	2	7	7	0				
13	7	2	7	0				
14	2	8	8	0				
15	8	2	8	0				
16	3	5	9	0				
17	5	3	9	0				
18	3	6	1	0				
19	6	3	1	1				
20	3	7	9	0				
21	7	3	9	0				
22	4	5	2	1				
23	5	4	2	0				
24	4	8	4	1				
25	8	4	4	0				

最小全域木問題をgurobiで解く(1)

- cplexの「モデルファイル(*.mod)」と「データファイル(*.dat)」を使って「lpファイル(*.lp)」を生成する
  - 例) モデルファイル [mst.mod], データファイル [mstex1.dat]
    → 生成する lpファイル [mstex1.lp]
  - [Win]+[R] キー で [ファイル名を指定して実行] d-boxを起動する
    - 枠内で cmd [Enter]
  - コマンドプロンプト command prompt のウィンドウ(黒い画面)が起動する
  - 以降, コマンドプロンプト内でコマンド(命令文)を打って順次命令を実行する

(1) モデルファイルとデータファイルがあるフォルダに移動する

```
cd [フォルダへのパス]  [Enter]
```

(2) 以下のコマンドを実行する

```
oplrun -e mstex1.lp mst.mod mstex1.dat  [Enter]
```

  - この結果, モデルファイル [mst.mod] とデータファイル [mstex1.dat] と同じフォルダ内に, lpファイル [mstex1.lp] が出来る(※確認すること)

最小全域木問題をgurobiで解く(1)

- gurobi を起動して問題を解き，最適解を得る
  - コマンドプロンプトで，以下の命令文を打って gurobi を起動する

```
gurobi  [Enter]
```

  - 起動した gurobi 内で，順次，以下の命令文を打って問題を解いていく

(1) 問題を記述してある lpファイル(mstex1.lp)を読み込み，model へセット

```
model = read("mstex1.lp")  [Enter]
```

(2) 解く(最適化計算を開始する)　※読込に失敗しているとエラーとなる

```
model.optimize()  [Enter]
```

(3) 最適解を表示する　※最適解が求まっていない場合はエラーとなる

```
model.printAttr('X')  [Enter]
```

(4) 最適値(目的関数値)を表示する　※同上

```
model.ObjVal  [Enter]
```

(5) 最適解をファイル(*.sol)に出力する　※ファイル名は好きに

```
model.write("mstex1.sol")  [Enter]
```

最小全域木問題をgurobiで解く(1)

- gurobi のその他，知っておくと便利な命令文
  - いずれも gurobi を起動して，gurobi内で実行する

    (a) ヘルプを表示する

```
help()   [Enter]
```

    (b) 全ての最適解(値が0の解)を表示する

```
for v in model.getVar() :    [Enter]
    print( v.VarName, ":", v.X)   [Enter]
```

  - 最適解を表示する命令文「m.printAttr('X')」は，値が0となる解は表示しない
  - 2行目の print 文は，必ず字下げ(インデント)して書くこと(Pythonの文法)
  - 字下げは[Tab]キーを使うと良い(※面倒でなければ，半角スペースでも可)
  - model.getVar() でモデルから変数Var(variableの頭3文字) を get する命令
  - get した各変数をインデックス v として，for文で繰り返す(2行目を繰り返す)
  - v.VarName は，ゲットした各変数の「名称」を意味する予約語
  - v.X は，ゲットした各変数の「値」を意味する予約語
  - 以上より，各変数を1つずつ「名称 : 値」の形で画面に表示(print)する

最小全域木問題をgurobiで解く(2)

1つのファイル「mst.py」に
①②③の順に記述して保存

- 問題(ex1)を python & gurobi で記述(mst.py)

①

```
# coding: Shift_JIS
from gurobipy import *

# ######## 例題設定 ########
def make_data_ex1():
    V = [1,2,3,4,5,6,7,8,9,10]
    E,c = multidict({(1,2):3,(1,3):5,(1,5):9,
(1,6):1,(2,3):2,(3,2):2,(2,4):4,(4,2):4,(2,5):6,(5,2)
:6,(2,7):7,(7,2):7,(2,8):8,(8,2):8,(3,5):9,(5,3):9,(3,
6):1,(6,3):1,(3,7):9,(7,3):9,(4,5):2,(5,4):2,(4,8):4,
(8,4):4,(4,9):9,(9,4):9,(4,10):3,(10,4):3,(5,8):7,(8,
5):7,(5,9):8,(9,5):8,(5,10):4,(10,5):4,(6,7):2,(7,6):
2,(6,9):3,(9,6):3,(7,9):5,(9,7):5,(7,10):5,(10,7):5,(
8,9):7,(9,8):7})
    return V,E,c
```

②

```
# ######## 定式化 ########
def mst(V,E,c):
    mod = Model("minimum spanning tree problem")

    # 変数設定
    x,u = {},{}
    for i in V:
        u[i] = mod.addVar(vtype="I", name="u(%s)" % i)
    for (i,j) in E:
        x[i,j] = mod.addVar(vtype="B", name="x(%s,%s)" % (i,j))
    mod.update()

    # 制約条件の設定
    for v in range(2,len(V)+1):
        mod.addConstr(quicksum(x[i,j] for (i,j) in E if j==v) == 1)
        mod.addConstr(u[v] >= 1)
        mod.addConstr(u[v] <= len(V)-1)
    mod.addConstr(u[1] == 0)
    for (i,j) in E:
        mod.addConstr(u[i] + 1 <= u[j] + (len(V)-1)*(1-x[i,j]))

    # 目的関数の設定
    mod.setObjective(quicksum(c[i,j]*x[i,j] for (i,j) in E), GRB.MINIMIZE)
    mod.update()
    mod.__data = x,u
    return mod
```

③

```
# ######## 実行 ########
if __name__=="__main__":
    V,E,c = make_data_ex1()          # データの
    mod = mst(V,E,c)                 # モデルの
    mod.write("mstex1.lp")           # lpファイル
    mod.optimize()                   # 最適化実
    print("¥n optimal value = ", mod.ObjVal)
    mod.printAttr('X')               # 最適解の
    mod.write("mstex1.sol")          # 最適解を
```

最小全域木問題をgurobiで解く(2)

- Pythonファイル(mst.py)をgurobi上で実行し，解く
  - [Win]+[R] キー で [ファイル名を指定して実行] d-boxを起動する
    - 枠内で cmd [Enter]
  - コマンドプロンプト command prompt のウィンドウ(黒い画面)が起動する
  - コマンドプロンプト内でコマンド(命令文)を打って順次命令を実行する

(1) 実行ファイルがあるフォルダに移動する

```
cd [フォルダへのパス] [Enter]
```

(2) 以下の命令文を打って gurobi を起動する

```
gurobi [Enter]
```

  - 起動した gurobi 内で，以下の命令文を打って問題を解く

```
gurobi> exec( open("mst.py").read() ) [Enter]
```

※python3系の場合

※python2系の場合の命令文は以下

```
gurobi> execfile("mst.py") [Enter]
```

最小全域木問題をgurobiで解く(2)

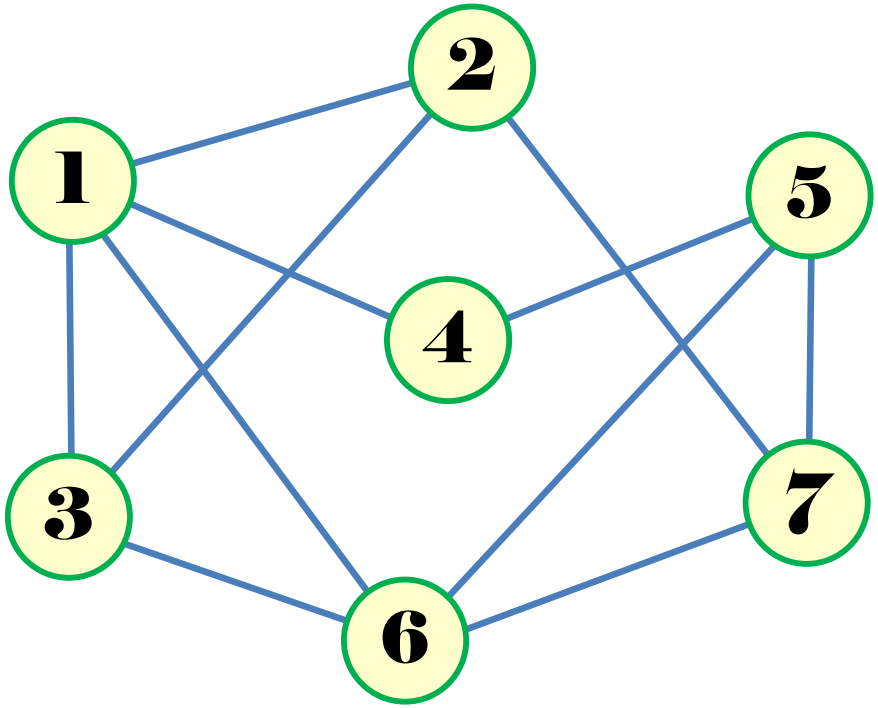
- 実行結果

```
gurobi> exec(open("mst.py").read())
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 10 physical cores, 20 logical processors, using up to 20 threads
Optimize a model with 72 rows, 54 columns and 195 nonzeros
Model fingerprint: 0xbb981479
Variable types: 0 continuous, 54 integer (44 binary)
Coefficient statistics:
  Matrix range     [1e+00, 9e+00]
  Objective range  [1e+00, 9e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 9e+00]
Found heuristic solution: objective 44.0000000
Presolve removed 35 rows and 16 columns
Presolve time: 0.00s
Presolved: 37 rows, 38 columns, 116 nonzeros
Variable types: 0 continuous, 38 integer (30 binary)
Found heuristic solution: objective 27.0000000

Root relaxation: objective 2.022222e+01, 13 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   20.22222    0    2   27.00000   20.22222  25.1%     -    0s
H    0     0                       25.0000000   20.22222  19.1%     -    0s
H    0     0                       24.0000000   20.22222  15.7%     -    0s
H    0     0                       22.0000000   20.22222  8.08%     -    0s
     0     0   20.22222    0    2   22.00000   20.22222  8.08%     -    0s

Explored 1 nodes (13 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 20 (of 20 available processors)

Solution count 5: 22 24 25 ... 44

Optimal solution found (tolerance 1.00e-04)
Best objective 2.200000000000e+01, best bound 2.200000000000e+01, gap 0.0000%
```

```
optimal value =  22.0

   Variable            X
-----------------------
       u(2)            7
       u(3)            2
       u(4)            8
       u(5)            9
       u(6)            1
       u(7)            2
       u(8)            9
       u(9)            9
      u(10)            9
     x(1,6)            1
     x(3,2)            1
     x(2,4)            1
     x(6,3)            1
     x(4,5)            1
     x(4,8)            1
    x(4,10)            1
     x(6,7)            1
     x(6,9)            1
gurobi>
```

【演習】最小全域木問題をCPLEXで解く

- ex2) グラフ $G = (V, E)$
  - 点集合 $V = \{1,2,3,4,5,6,7\}$,
  - 枝集合 $E = \{(1,2), (1,3), (1,4), (1,6), (2,3), (2,3), (2,7), (3,6), (4,5), (5,6), (5,7), (6,7)\}$



- 問
  1. $|V| = ?$ $|E| = ?$
  2. 無向グラフの枝集合から有向グラフの枝集合をつくれ
  3. 例1と同様に変数を設定し，定式化せよ
  4. 整数計画ソルバー（cplex）を用いて，最大安定集合を求めよ
  5. oplrun を使って，mod file / dat file から lp file を作れ
  6. 整数計画ソルバー（gurobi）で5のlp file を解き，最大安定集合を求めよ
  7. 整数計画ソルバー（gurobi）とpython で解き，最大安定集合を求めよ
  8. 結果を networkx でグラフ描画せよ

【演習】最小全域木問題をCPLEXで解く

- ex3)
  - ランダムグラフ $G = (V, E)$ で問題を作る（python/networkx等を利用）
    - 点集合の要素数 $|V|$ を適当に設定（$n = 5$〜20 程度）
    - 枝集合 E の密度を適当に設定（0.0〜1.0）

- 問
  1. $|V| = ?$　$|E| = ?$
  2. 無向グラフの枝集合から有向グラフの枝集合をつくれ
  3. 例1と同様に変数を設定し，定式化せよ
  4. 整数計画ソルバー（cplex）を用いて，最大安定集合を求めよ
  5. oplrun を使って，mod file / dat file から lp file を作れ
  6. 整数計画ソルバー（gurobi）で5のlp file を解き，最大安定集合を求めよ
  7. 整数計画ソルバー（gurobi）とpython で解き，最大安定集合を求めよ
  8. 結果を networkx でグラフ描画せよ