|  |  |
| --- | --- |
| var Alma >=0, integer; var Citrom >=0, integer;  s.t. C: 100\*Alma + 300\*Citrom >= 500;  s.t. F: 0.5\*Alma + 0.2\*Citrom >= 1.2;  s.t. Kcal: 80\*Alma + 50\*Citrom <= 400;  minimize kiadas: (400\*Alma + 600\*Citrom) \* 0.1; | var ing >= 0, integer; var szoknya >= 0, integer;  s.t. munkaido: 5\*ing + 2\*szoknya <= 10;  s.t. anyagfelh: 3\*ing + 4\*szoknya <= 12;  s.t. darabszam: ing + szoknya >= 4;  maximize bevetel: 4\*ing + 3\*szoknya; |
| var T1 >= 0, integer; var T2 >= 0, integer;  s.t. Gep\_1: 3\*T1 + 4\*T2 <= 130;  s.t. Gep\_2: 4\*T1 + 6\*T2 <= 200;  s.t. Munkas: 1\*T1 + 2\*T2 = 60;  maximize bevetel: 8\*T1 + 10\*T2; | var T1 >= 0, integer; var T2 >= 0, integer;  s.t. gep\_1: (1\*1 + 2\*1) \*T1 + (1\*0 + 1\*1) \*T2 <= 240;  s.t. gep\_2: (1\*7 + 2\*1) \*T1 + (1\*1 + 1\*1) \*T2 <= 630;  s.t. szerelde: 2\*T1 + 1\*T2 <= 220;  maximize bevetel: 27\*T1 + 8\*T2; |
| var T1 >= 0, integer; var T2 >= 0, integer;  var T3 >= 0, integer;  s.t. eszterga: 2\*T1 + 1\*T2 + 1\*T3 <= 80;  s.t. marogep: 2\*T1 + 1\*T2 + 1\*T3 <= 70;  s.t. bevetel: 3\*T1 + 1\*T2 + 2\*T3 >= 100;  minimize anyagkoltseg: 0.8\*T1 + 0.2\*T2 + 0.5\*T3; | var lekvar >= 0, integer; var befott >= 0, integer;  s.t. gyumi: 80\*lekvar + 40\*befott <= 12000;  s.t. cukor: 40\*lekvar + 40\*befott <= 8000;  maximize bevetel: 100\*lekvar + 80\*befott; |
| var x1 >= 0, integer; var x2 >= 0, integer;  var x3 >= 0, integer; var x4 >= 0, integer;  s.t. feherje: 50 \* x1 + 100\*x2 + 30\*x3 + 10\*x4 >= 120;  s.t. szenhidrat: 300 >= 100 \* x1 + 10\*x2 + 90\*x3 + 100\*x4 >= 200;  s.t. zsir: 60 >= 5 \* x1 + 10\*x2 + 6\*x3 + 20\*x4 >= 50;  minimize koltseg: 500 \*x1 + 1500\*x2 + 600\*x3 + 90\*x4;  solve;  printf "A koltseg: %d\n", koltseg;  printf "Az egyik kaja: %d\n", x1; printf "Az egyik kaja: %d\n", x2;  printf "Az egyik kaja: %d\n", x3; printf "Az egyik kaja: %d\n", x4; | var x12, binary; var x24, binary; var x3, binary;  var x57, binary; var x45, binary; var x78, binary;  s.t. nap1: x12 <= 1;  s.t. nap2: x12 + x24 <= 1;  s.t. nap3: x24 + x3 <= 1;  s.t. nap4: x24 + x45 <= 1;  s.t. nap5: x57 + x45 <= 1;  s.t. nap6: x57 <= 1;  s.t. nap7: x57 + x78 <= 1;  s.t. nap8: x78 <= 1;  maximize kurzusok: x12 + x24 + x3 + x57 + x45 + x78; |
| var x1 >= 0; var x2 >= 0; var x3 >= 0;  var x4 >= 0; var y1 >= 0; var y2 >= 0;  var y3 >= 0;  s.t. fst: 1\*x1 + 5\*x2 +4\*x3 +2\*x4 +y1 = 10;  s.t. snd: 3\*x1 + 6\*x2 +2\*x3 +10\*x4 +y2 = 20;  s.t. thr: 4\*x1 + 8\*x2 +10\*x3 +12\*x4 +y3 = 30;  maximize Fx: 5\*x1 + 2\*x2 + 6\*x3 +8\*x4; | var x21 >= 0, integer; var x7 >= 0, integer;  var x5 >= 0, integer; var x1 >= 0, integer;  var x42 >= 0, integer;  s.t. osszeg: 21\*x21 + 7\*x7 + 5\*x5 + x1 + 42\*x42 = 1234;  minimize darabszam: x21 + x7 + x5 + x1 + x42; solve;  printf "A darabszám: %d \n", darabszam;  printf "21-ből ennyi: %d \n", x21; printf "7-ből ennyi: %d \n", x7;  printf "5-ből ennyi: %d \n", x5; printf "1-ből ennyi: %d \n", x1;  printf "42-ből ennyi: %d \n", x42;  printf "És akkor egybe: %d %d %d %d %d\n", x21, x7, x5, x1, x42; |
| var x1, binary; var x2, binary; var x3, binary;  var x4, binary; var x5, binary; var y1, binary; #2  var y2, binary; #3 var y3, binary; #1 var y4, binary; #4  var y5, binary; #5 var y6, binary; #6  s.t. cs\_miki: x2 + x4 + x5 - y1 = 1; s.t. narancs: x1 + x4 - y2 = 1;  s.t. mogyaro: x1 + x2 -y3 = 1; s.t. virgacs: x3 + x5 -y4 = 1;  s.t. kindert: x3 + x4 -y5 = 1; s.t. gumicuk: x2 + x3 -y6 = 1;  minimize koltseg: (5\*x1+ 10\*x2 + 3\*x3 + 4\*x4 + 6\*x5) + (2\*y1 + 3\*y2 + 1\*y3 + 4\*y4 + 5\*y5 + 6\*y6); | var x1, binary; var x2, binary; var x3, binary;  var x4, binary; var x5, binary;  s.t. cs\_miki: x2 + x4 + x5 >= 1;  s.t. narancs: x1 + x4 >= 1;  s.t. mogyaro: x1 + x2 >= 1;  s.t. virgacs: x3 + x5 >= 1;  s.t. kindert: x3 + x4 >= 1;  s.t. gumicuk: x2 + x3 >= 1;  minimize koltseg: (5\*x1+ 10\*x2 + 3\*x3 + 4\*x4 + 6\*x5); |
| set Eszkozok;  param Suly{Eszkozok};  param Ertek{Eszkozok};  param MaxSuly;  var x{Eszkozok} binary;  s.t. korlat: sum{i in Eszkozok} Suly[i] \* x[i] <= MaxSuly;  maximize maxErtek: sum{i in Eszkozok} Ertek[i] \* x[i]; end; | data;  set Eszkozok := ko papir ollo;  param Suly := ko 4  papir 5  ollo 3;  param Ertek := ko 10  papir 15  ollo 6;    param MaxSuly := 10; end; |
| set Gyarak;  set Boltok;  param GyarakTermeles {Gyarak};  param BoltokFogyasztas {Boltok};  param Koltsegek{Gyarak, Boltok};  var x{Gyarak, Boltok} >= 0;  s.t. gyar{i in Gyarak}: sum{j in Boltok} x[i,j] = GyarakTermeles[i];  s.t. bolt{j in Boltok}: sum{i in Gyarak} x[i,j] = BoltokFogyasztas[j];  minimize OsszKoltseg: sum{i in Gyarak} sum{j in Boltok} Koltsegek[i,j] \* x[i,j];  end; | data;  set Gyarak := Budapest Veszprem Gyor Szekesfehervar;  set Boltok := Komarom Szeged Debrecen;  param GyarakTermeles := Budapest 1000  Veszprem 500  Gyor 600  Szekesfehervar 400;  param BoltokFogyasztas := Komarom 1100  Szeged 700  Debrecen 700;    param Koltsegek : Komarom| Szeged | Debrecen :=  Budapest 12 5 5  Veszprem 5 4 3  Gyor 2 10 12  Szekesfehervar 3 6 7;  end; |
| set TABLETTAK;  param ar{TABLETTAK};  param hatasfok{TABLETTAK};  param adagolas\_limit;  var adagolas{TABLETTAK}>=0 integer;  s.t. kettes\_szint: sum{t in TABLETTAK} adagolas[t] \* hatasfok[t] >= 51;  s.t. tuladagolas: sum{t in TABLETTAK} adagolas[t] <= adagolas\_limit;  s.t. arany\_1: adagolas['Spenotos'] = adagolas['Epres'];  s.t. arany\_2: adagolas['Spenotos'] = adagolas['Kiwis'];  minimize Kiadas: sum{t in TABLETTAK} adagolas[t] \* ar[t];  end; | data;  set TABLETTAK:=Spenotos  Epres  Kiwis;  param ar:= Spenotos 2000  Epres 1500  Kiwis 1000;    param hatasfok:= Spenotos 10  Epres 7  Kiwis 10;    param adagolas\_limit := 8; |
| var spenot >= 0, integer; var eper >= 0, integer;  s.t. mertek: spenot + eper <= 8;  s.t. osszetetel: spenot <= eper;  s.t. siker: 10\*spenot + 7\*eper >= 51;  minimize koltseg: 2000\*spenot + 1500\*eper; | var AD >= 0; var AE >= 0; var BD >= 0;  var BF >= 0; var CD >= 0; var CF >= 0;  var DE >= 0; var DF >= 0;  s.t. A: AD + AE = 100; s.t. B: BD + BF = 150;  s.t. C: CD + CF = 200; s.t. D: AD + BD + CD - 50 = DE + DF;  s.t. E: AE + DE = 150; s.t. F: DF + BF + CF = 250;  minimize szallitas\_koltseg: 5\*AD + 3\*AE + 2\*BD + 2\*BF + 10\*CD + 4\*CF + 3\*DE + 4\*DF; |
| 1. Cél fv  2. melyik negyed?  3. s.t. -> egyenlőtlenség egyenletre  4. X: y-t 0-ra -> X metszet  5. Y: x-t 0-ra -> Y metszet  6. s.t.-ok alatt/felett  7. s.t.-ok metszéspontja -> 2 egyenletes egyenletrendszer  8. corner pointok -> max/min kiválasztása |  |