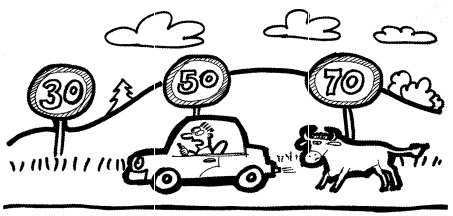
INFORMATIKA

Omezovač rychlosti (Úlohy z MO – kategorie P, 25. část)

PAVEL TÖPFER

Matematicko-fyzikální fakulta UK, Praha



Náměty soutěžních úloh právě skončeného 59. ročníku Matematické olympiády – kategorie P (školní rok 2009/2010) připravili slovenští organizátoři olympiády z Fakulty matematiky, fyziky a informatiky Univerzity Komenského v Bratislavě. V našem seriálu zajímavých úloh z olympiády si tentokrát rozebereme jednu úlohu z krajského kola soutěže, kterou bychom mohli označit jako grafovou a zároveň jako optimalizační. Ačkoliv se jedná o úlohu netradiční, k jejímu vyřešení nebudeme potřebovat žádné speciální znalosti, ale úplně vystačíme s prostou logickou úvahou a se základními grafovými algoritmy. Začneme jako obvykle zadáním soutěžní úlohy, které jsme zde oproti původní verzi trochu upravili a zkrátili, aniž by se ovšem jakkoliv změnil původní smysl úlohy.

⋆⋆⋆

Společnost Expresní pošta doručuje zásilky po celé Evropě. V poslední době ale její řidiči často přehlíželi dopravní značky a dostávali pokuty za překročení maximální povolené rychlosti. Ředitel společnosti proto rozhodl, že nechá do každého auta namontovat *omezovač rychlosti*. Ten fun-

guje tak, že si na něm řidič před jízdou nastaví maximální přípustnou rychlost a přístroj se pak automaticky postará o to, aby auto během celé jízdy nikdy tuto rychlost nepřekročilo. Ředitel společnosti navíc vydal nařízení, že si řidič musí před každou jízdou nastavit takové omezení rychlosti, aby nikde na své trase nepřekročil žádnou maximální povolenou rychlost.

Soutěžní úloha

Je dána silniční síť, po níž jezdí řidiči společnosti Expresní pošta. Tato silniční síť obsahuje N měst, mezi nimiž vede celkem M různých silnic. Každá silnice spojuje dvě města, přičemž silnice se mimo města kříží pouze mimoúrovňově (tzn. mimo města nelze odbočit z jedné silnice na jinou). Pro každou silnici známe její délku (v kilometrech) a maximální povolenou rychlost (v kilometrech za hodinu). Pro danou dvojici měst x, y může existovat více způsobů, jak lze po silnicích dojet z města x do města y. Napište program, který pro všechny dvojice měst x, y určí minimální čas potřebný na cestu z města x do města y při použití omezovače rychlosti a dodržení nařízení ředitele společnosti.

Formát vstupu

První řádek obsahuje dvě kladná celá čísla N, M ($1 \leq N \leq 50$, $1 \leq M \leq 1000$) – počet měst a počet silnic mezi nimi. Jednotlivá města jsou na vstupu označena čísly 1 až N. Každý z následujících M řádků popisuje jednu silnici. Obsahuje čtyři čísla i, j, d, m, která udávají, že silnice spojující města i, j má délku d kilometrů a maximální povolenou rychlost m kilometrů za hodinu. Všechny silnice jsou obousměrné. Mezi dvěma městy může vést více silnic různé délky a s různou maximální povolenou rychlostí. Můžete předpokládat, že mezi každou dvojicí měst existuje aspoň jedna trasa (která může být tvořena více navazujícími silnicemi).

Všechny vzdálenosti a rychlosti jsou na vstupu uvedeny s přesností nejvýše na tři desetinná místa. Pro každou vzdálenost d platí $1 \leq d \leq$ ≤ 1000000, pro každou maximální povolenou rychlost m platí $5 \leq m \leq$ ≤ 100000.

Formát výstupu

Výstup bude tvořen N řádky, z nichž každý obsahuje N čísel. Číslo v i-tém řádku a j-tém sloupci určuje minimální čas (v hodinách) potřebný na jízdu mezi městy i, j. Výsledek uveďte s přesností na tři desetinná místa.

Poznámka

Při práci s reálnými čísly v počítači mohou vznikat zaokrouhlovací chyby. Tuto skutečnost můžete ve svém řešení ignorovat – postupujte, jako kdyby všechny výpočty, které provádíte, byly přesné.

⋆⋆⋆

Na konkrétním příkladu vstupních dat a jim odpovídajících výsledků si objasníme zadání a zároveň ukážeme, čím je tato úloha zajímavá a na co musíme dát pozor při jejím řešení.

Vstup

```
4 4
1 2 40.0 60.0
1 3 20.0 100.0
2 3 40.0 120.0
2 4 25.0 50.0
```

Výstup

```
0.000 0.600 0.200 1.300
0.600 0.000 0.333 0.500
0.200 0.333 0.000 1.300
1.300 0.500 1.300 0.000
```

Podle vstupních dat si sami nakreslete jednoduché schéma zadané silniční sítě. V uvedeném příkladu se z města 1 do města 2 nejrychleji dostaneme přes město 3. Ujedeme při tom celkem 60 kilometrů rychlostí 100 km/h, takže doba jízdy bude 0,6 hod. Existuje kratší přímé spojení mezi městy 1 a 2, které má délku pouze 40 km, ale omezení rychlosti na této silnici na 60 km/h způsobí delší dobu jízdy 0,667 hod. Je tedy zřejmé, že nejkratší cesta mezi městy x, y nemusí být vždy časově nejvýhodnější. Úlohu proto nemůžeme jednoduše převést na pouhé hledání nejkratších cest mezi všemi dvojicemi měst, což bychom uměli vyřešit pomocí standardních grafových algoritmů (Dijkstrův algoritmus, Floydův-Warshallův algoritmus – viz např. [1]).

Ve výše uvedeném příkladu si dále všimněte, že nejrychlejší cesta z města 1 do města 4 vede po trase 1-2-4, nikoliv 1-3-2-4. Vzhledem k omezení rychlosti na silnici 2-4 na 50 km/h musíme takto nastavit omezovač

rychlosti pro celou jízdu a již se nám proto nevyplatí optimalizovat cestu z města 1 do města 2 objížďkou přes město 3, jako tomu bylo v předchozím případě. Vidíme tedy, že pokud vede nejrychlejší cesta z města x do města y přes město z, nemůžeme ji získat prostým složením nejrychlejší cesty z města x do města z a nejrychlejší cesty z města z do města y. Úlohu proto nemůžeme řešit ani žádným jednoduchým dynamickým programováním, které by postupovalo s rostoucím počtem měst, jimiž na zvolené trase projíždíme.

Pro všechny další úvahy při řešení úlohy bude důležité následující pozorování. Ať si zvolíme jakoukoliv trasu vedoucí z města x do města y, optimální nastavení omezovače rychlosti bude rovno minimu z maximální rychlosti povolené na silnicích, které tvoří zvolenou trasu. Vyšší rychlost na omezovači nastavit nesmíme, nastavit nižší rychlost se nám nevyplatí. Pro dosažení nejkratší doby jízdy mezi kterýmikoliv dvěma městy má proto smysl nastavovat omezovač rychlosti jedině na některou z rychlostí, které jsou zadány na vstupu.

Nejrychlejší cestu z města x do města y bychom mohli teoreticky získat tak, že bychom nalezli všechny cesty vedoucí z x do y, pro každou z nich bychom určili nastavení omezovače rychlosti (minimum z maximálních povolených rychlostí na silnicích tvořících tuto cestu) a její délku (součet délek silnic tvořících tuto cestu) a spočítali bychom dobu jízdy. Tento postup je ale velmi neefektivní, v případě husté silniční sítě bude počet existujících cest exponenciálně velký vzhledem k N.

Výrazně lepší řešení získáme vhodným využitím některého ze standardních grafových algoritmů. Uvažovaná silniční síť představuje hranově ohodnocený neorientovaný graf G, vrcholy grafu odpovídají městům (je jich N) a hrany exitujícím silnicím (těch je M). Již jsme konstatovali, že omezovač rychlosti bude vždy nastaven na některou z rychlostí uvedených na vstupu. Těchto možných rychlostí není mnoho – nejvýše tolik, kolik je silnic. Nabízí se tedy možnost vyzkoušet postupně všechna možná nastavení omezovače rychlosti a pro každou takovou rychlost v určit, za jaký čas se dokážeme dostat z města x do města y. Označme si symbolem G_v podgraf původního grafu, který obsahuje jen ty hrany, na nichž je rychlost v ještě povolena. Máme-li tedy nastaven omezovač rychlosti na hodnotu v, můžeme jet pouze po hranách grafu G_v. Když už máme určenou rychlost jízdy, nejlepší dobu jízdy dosáhneme tehdy, pokud v grafu G_v zvolíme nejkratší možnou cestu (v kilometrech) vedoucí z města x do města y.

Celý postup řešení ještě jednou shrneme: Pro každou rychlost v ze

vstupu vytvoříme graf G_v. V něm určíme délky (v kilometrech) nejkratších cest mezi všemi dvojicemi měst x, y a označíme je $D_v(x, y)$. Výsledné optimální doby jízdy pro rychlost v mají hodnotu $D_v(x, y)/v$. Pro každou dvojici měst x, y zvlášť pak stačí určit, při kterém v nabývá výraz $D_v(x, y)/v$ minimální hodnoty. Tato minimální hodnota je hledaným výsledkem.

Zbývá ukázat, jakou časovou složitost má popsané řešení. Podle našeho úvodního pozorování budeme jako hodnoty v uvažovat všechny rychlosti zadané na vstupu – těch je nejvýše M různých. Graf G_v sestrojíme snadno v čase $O(M)$ tak, že projdeme všechny hrany původního grafu. Pro určení délky nejkratší cesty v grafu G_v mezi každou dvojicí vrcholů se nabízejí dva možné přístupy:

- Floydův-Warshallův algoritmus, který tento problém řeší v čase $O(N^3)$
- N-krát spuštěný Dijkstrův algoritmus (pro každý výchozí vrchol), čímž dojdeme opět k časové složitosti $O(N^3)$, nebo v případě využití haldy získáme řešení v čase $O(NM \log N)$.

Celková časová složitost řešení je tedy $O(MN^3)$, příp. $O(NM^2 \log N)$, paměťová složitost algoritmu je $O(N^2)$.

Právě popsané řešení ale ještě není nejlepší možné. Ukážeme si nyní jiný postup, který využívá trochu odlišným způsobem podobnou myšlenku, jako již zmíněný Floydův-Warshallův algoritmus (popis Floydova-Warshallova algoritmu najdete třeba v učebnici [1] nebo ve studijním textu [2]). Nejprve seřadíme všechny hrany v grafu G do posloupnosti $e_1, e_2, \ldots, e_M$ tak, aby pro jejich maximální povolené rychlosti platilo $v_1 = v_2 = v_3 = \cdots = v_M$. Úlohu nyní vyřešíme metodou dynamického programování. Postupně pro každé k od 1 do M spočítáme všechny hodnoty $d_k(x, y)$, které představují délku nejkratší cesty (v kilometrech) mezi městy x, y, jestliže se smíme pohybovat pouze po hranách $e_1, e_2, \ldots, e_k$. Pokud taková cesta neexistuje, položíme $d_k(x, y)$ rovno nekonečnu. Podle definice z předcházejícího řešení můžeme také říci, že $d_k(x, y)$ je délka nejkratší cesty mezi městy x, y v grafu G_{v_k}. K vyřešení úlohy stačí nalézt hodnoty $d_k(x, y)$ pro každé $k = 1, 2, \ldots, M$. Hodnota výrazu $d_k(x, y)/v_k$ pak určuje příslušnou optimální dobu jízdy pro rychlost v_k. Podobně jako v předchozím řešení získáme hledaný výsledek pro každou dvojici měst x, y jako minimum z hodnot $d_k(x, y)/v_k$ získaných pro $k = 1, 2, \ldots, M$.

Ukážeme si nyní, jak lze rychle a elegantně spočítat hodnoty $d_k(x, y)$ na základě již známých hodnot $d_{k-1}(x, y)$. V k-tém kroku výpočtu chceme „zpřístupnit“ hranu e_k a přepočítat hodnoty $d_k(x, y)$ pro všechny dvojice

měst x, y. Při stanovení hodnoty $d_k(x, y)$ uvažujeme následující dvě možnosti:

1. V případě, že nejkratší cesta z města x do města y nepoužije hranu e_k, je
$$d_k(x, y) = d_{k-1}(x, y).$$

2. V opačném případě lze nejkratší cestu z x do y rozdělit na tři úseky: z vrcholu x půjdeme do některého vrcholu hrany e_k, projdeme touto hranou a z jejího druhého konce půjdeme do vrcholu y. V prvním i ve třetím úseku cesty se vyskytují pouze hrany s číslem menším než k, takže optimální délku těchto úseků už známe. Jestliže označíme u_{k1} a u_{k2} vrcholy spojené hranou e_k, potom platí buď
$$d_k(x, y) = d_{k-1}(x, uk1) + |e_k| + d_{k-1}(u_{k2}, y),$$
nebo
$$d_k(x, y) = d_{k-1}(x, u_{k2}) + |e_k| + d_{k-1}(u_{k1}, y),$$
podle toho, kterým směrem naše trasa prochází hranou e_k.

Můžeme si zvolit, zda hranu e_k použijeme nebo ne, a pokud ano, tak v kterém směru. Vybereme si samozřejmě nejlepší z uvedených tří možností. Proto je hodnota $d_k(x, y)$ rovna minimu z uvedených tří možností. Tím jsme získali rekurzivní vztah pro výpočet hodnot $d_k(x, y)$ pomocí hodnot $d_{k-1}(x, y)$.

V každém z M kroků výpočtu (pro $k = 1, 2, \ldots, M$) spočítáme každou z N^2 hodnot (pro všechny dvojice $x, y = 1, 2, \ldots, N$) podle uvedeného rekurzivního vztahu v konstantním čase. Toto řešení má proto časovou složitost $O(MN^2)$. K tomu je ještě třeba připočítat čas potřebný na uspořádání hran podle maximální povolené rychlosti – to můžeme provést v čase $O(M \log M)$. Celková časová složitost popsaného algoritmu je tedy $O(MN^2 + M \log M)$.

Pro dosažení nízké paměťové složitosti si stačí uvědomit, že hodnoty d_k závisejí pouze na hodnotách d_{k-1}, nikoliv na hodnotách ze starších kroků výpočtu. Vystačíme proto s pamětí velikosti $O(N^2)$, starší hodnoty si nemusíme pamatovat.

Na závěr připojujeme ukázkový program v Pascalu, který demonstruje, jak lze právě popsaný algoritmus naprogramovat. Pro uspořádání hran

podle maximální povolené rychlosti zde použijeme jednoduchý třídicí algoritmus přímého výběru, který má sice časovou složitost $O(M^2)$, ale jeho zápis je zato krátký a přehledný. Mohli bychom ho samozřejmě v programu nahradit nějakým efektivnějším třídicím algoritmem.

Pokud bychom psali program přesně podle výše uvedeného algoritmu, potřebovali bychom v něm mít dva exempláře pole na ukládání vzdáleností měst. V každém kroku výpočtu bychom měli v jednom poli uloženy již spočítané hodnoty $d_{k-1}(x, y)$ a do druhého bychom ukládali nově počítané hodnoty $d_k(x, y)$. Role obou polí by se pravidelně střídaly. Ve skutečnosti ale můžeme napsat program ještě úsporněji – vystačíme s jediným polem, v němž budeme v k-tém kroku výpočtu uložené hodnoty $d_{k-1}(x, y)$ přímo nahrazovat novými hodnotami $d_k(x, y)$. Při postupném výpočtu prvků d_k se nám pak ovšem může stát, že místo „správné“ hodnoty d_{k-1} někde použijeme již upravenou novou hodnotu d_k. Rozborem možných situací snadno zjistíte, že i v takovém případě získáme po k-tém kroku výpočtu správné hodnoty $d_k(x, y)$.

```
program OmezovacRychlosti;

const MaxN = 50;         {maximální počet měst = vrcholů}
      MaxM = 1000;       {maximální počet silnic = hran}
      NEKONECNO = 1e30; {\uv{nekonečná} vzdálenost měst}

type hrana = record
               i, j: integer; {koncové vrcholy hrany}
               d, m: real;    {vzdálenost, max. povolená
                               rychlost}
             end;

var n: integer;          {počet měst = vrcholů}
    m: integer;          {počet silnic = hran}
    h: array[1..MaxM] of hrana;    {seznam všech hran}
    d: array[1..MaxN, 1..MaxN] of real;
                         {vzdálenosti při použití
                          prvních k hran}
    vysl: array[1..MaxN, 1..MaxN] of real;
                         {výsledné časy jízd mezi
                          městy}
```

```
    x, y, i, j, k: integer;
    z: hrana;

function min(a, b: real):real;
begin
if a<b then
  min:=a
else
  min:=b
end;

begin
{Načtení vstupních dat a inicializace polí:}
read(n, m);
for k:=1 to m do
  with h[k] do read(i, j, d, m);
for x:=1 to n do
  for y:=1 to n do
    if x=y then
      begin d[x,y]:=0; vysl[x,y]:=0 end
    else
      begin d[x,y]:=NEKONECNO; vysl[x,y]:=NEKONECNO end;

{Seřazení hran podle maximální povolené rychlosti sestupně.
 Zde je použit algoritmus přímého výběru:}

for i:=1 to m-1 do      {umístit hranu na pozici i}
  begin
  k:=i;
  for j:=i+1 to m do    {vyhledání maxima}
    if h[j].m > h[k].m then k:=j;
  if k > i then         {výměna hran s indexy i, k}
    begin z:=h[k]; h[k]:=h[i]; h[i]:=z end
  end;

{Vlastní výpočet vzdáleností a časů:}
for k:=1 to m do       {k kroků výpočtu -- povolíme
                        vždy hranu h[k]}
```

```
  for x:=1 to n do
    for y:=1 to n do
      begin
      d[x,y]:=min(d[x,y], d[x,h[k].i] + h[k].d +
              + d[h[k].j,y]);
      d[x,y]:=min(d[x,y], d[x,h[k].j] + h[k].d +
              + d[h[k].i,y]);
      vysl[x,y]:=min(vysl[x,y], d[x,y]/h[k].m)
      end;

{Výpis výsledků:}
for x:=1 to n do
  begin
  for y:=1 to n do write(vysl[x,y]:8:3);
  writeln
  end
end.
```

Literatura

[1] *Töpfer, P*: Algoritmy a programovací techniky, Prometheus, Praha 2007 (2. vydání)

[2] Korespondenční seminář z programování MFF UK, studijní materiály, http://ksp.mff.cuni.cz/tasks/21/cook5.html

Další možnosti Řešitele v Excelu

PETR EMANOVSKÝ

Přírodovědecká fakulta Univerzity Palackého v Olomouci

V článku [1] popisují autoři doplňkovou aplikaci MS Excelu zvanou *Řešitel* a na konkrétních příkladech ukazují její užitečnost při hledání nulových, maximálních a minimálních hodnot funkcí. Úlohy tohoto typu se často vyskytují v matematice, fyzice, ekonomii i v jiných oborech. Pro

středoškolské studenty je tento numerický nástroj vhodný například při řešení rovnic nebo optimalizačních úloh, a to zejména v situacích, kdy je „ruční“ řešení časově náročné nebo středoškolskými prostředky obtížně realizovatelné. *Řešitel* tedy představuje běžně dostupný software, který lze s výhodou používat v rámci výuky matematiky a dalších předmětů na střední škole.

Základní návod pro práci s *Řešitelem* je uveden v článku [1], případně lze nahlédnout například do publikací [2] nebo [3]. V předloženém článku ukážeme další využití *Řešitele*, konkrétně využití údajů tzv. citlivostní zprávy při řešení úloh lineárního programování.

1. Úlohy lineárního programování

Úlohy lineárního programování představují speciální případ optimalizačních úloh, při nichž je třeba nalézt maximální nebo minimální hodnotu nějaké lineární funkce, přičemž je třeba vyhovět všem omezujícím podmínkám, které mají tvar lineárních nerovnic nebo lineárních rovnic. Studenti středních škol se s jednoduchými úlohami lineárního programování mohou setkat v semináři z matematiky viz úloha o maximalizaci zisku v [1]. Ukažme si na této úloze a několika jejích modifikacích základní pojmy a postupy tzv. citlivostní analýzy lineárního programování (viz např. [3]), která zkoumá, jak se mění optimální řešení úlohy v závislosti na změnách vstupních dat modelu. K této analýze lze využít již zmiňované citlivostní zprávy *Řešitele*.

1.1 Příklad – maximalizace zisku

Papírenská firma vyrábí dva typy sešitů. Pro konkrétnost je označme jako A, resp. B. Při výrobě těchto sešitů jsou použity dva stroje – stroj na střihání papíru a stroj na sešívání listů. Sešit typu A se na střihacím stroji připravuje 2 minuty a sešit typu B se připravuje 6 minut. Na šicím stroji se sešit typu A sešívá 8 minut a sešit typu B 4 minuty. Protože výrobce najímá ke každému stroji pouze jednoho pracovníka, může každý z obou výrobních procesů trvat pouze 8 hodin denně, tj. 480 minut. Zisk z prodeje každého sešitu typu A činí 30 Kč a zisk z prodeje každého sešitu typu B 40 Kč. Kolik sešitů jednotlivých typů by měla firma vyrábět, aby bylo dosaženo maximálního zisku?

Jedná se o jednoduchou úlohu lineárního programování s matematickým modelem tvaru:

maximalizovat funkci

$$P(x, y) = 30x + 40y,$$

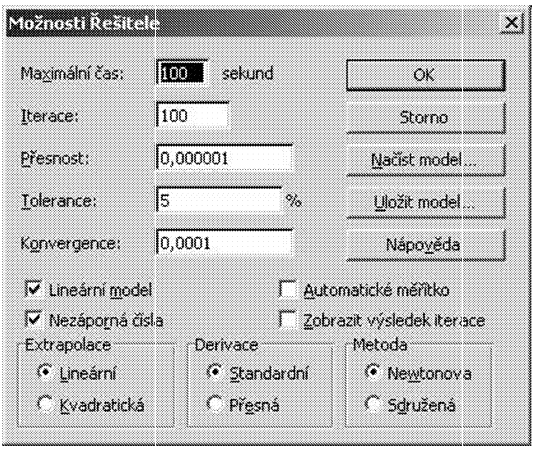
za podmínek

$$2x + 6y \leq 480$$
$$8x + 4y \leq 480$$
$$x \geq 0 \text{ a } y \geq 0$$

(viz [1]).

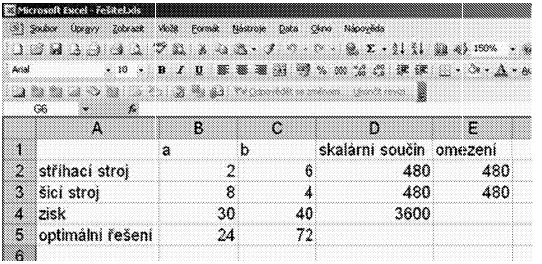
1.2 Řešení v Excelu

Pro řešení úloh lineárního programování se doporučuje nejprve nastavit některé parametry *Řešitele*. Toto nastavení lze provést v dialogovém okně **Možnosti Řešitele**, které je součástí okna **Parametry Řešitele** (obr. 1). Pro naše účely je důležité zejména vybrat položky **Lineární model** a **Nezáporná čísla**. Zapnutím dvoupolohového přepínače „Lineární model“ můžeme výrazně zkrátit dobu výpočtu a rovněž tím změníme podobu některých výstupních dat (např. tzv. citlivostní zprávy). Druhý dvoupolohový přepínač „Nezáporná čísla“ umožňuje automaticky zajistit podmínky nezápornosti, které pak již nemusíme zadávat spolu s ostatními omezujícími podmínkami modelu.



Obr. 1 Nastavení parametrů *Řešitele* pro úlohy lineárního programování

Pomocí postupu popsaného v [1] lze efektivně určit optimální řešení úlohy. V našem případě se jedná o hodnoty $x = 24$ a $y = 72$, což znamená, že je třeba vyrábět 24 kusů sešitů typu A a 72 kusů sešitů typu B za den. Maximální zisk pak bude činit 3600 Kč (obr. 2).

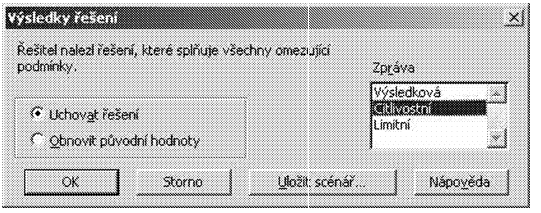


	A	B	C	D	E
1		a	b	skalární součin	omezení
2	střihací stroj	2	6	480	480
3	šicí stroj	8	4	480	480
4	zisk	30	40	3600	
5	optimální řešení	24	72		
6					

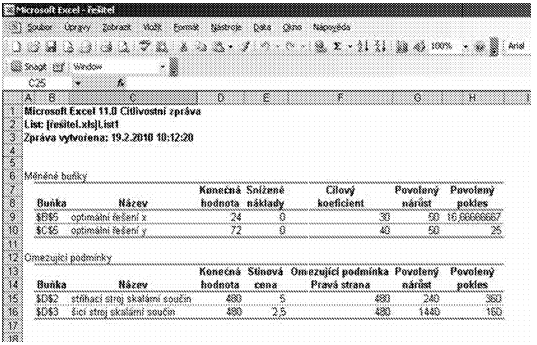
Obr. 2 Výstup MS Excelu po výpočtu optimálního řešení *Řešitelem*

2. Zprávy Řešitele

Po ukončení výpočtu *Řešitelem* má uživatel kromě základní podoby výstupu vypočtených hodnot možnost volby výstupu podrobnějších informací ve formě zpráv. *Řešitel* nabízí tři druhy zpráv – výsledkovou, citlivostní a limitní. Zvolíme-li požadovaný typ zprávy v dialogovém okně **Výsledky řešení** (obr. 3), je tato zpráva vygenerována do samostatného listu spreadsheet (obr. 4).



Obr. 3 Volba citlivostní zprávy v dialogovém okně „Výsledky řešení“



Microsoft Excel 11.0 Citlivostní zpráva
List: [řešitel.xls]List1
Zpráva vytvořena: 19.2.2010 10:12:20

Měněné buňky

Buňka	Název	Konečná hodnota	Snížené náklady	Cílový koeficient	Povolený nárůst	Povolený pokles
B5	optimální řešení x	24	0	30	50	16,66666667
C5	optimální řešení y	72	0	40	50	25

Omezující podmínky

Buňka	Název	Konečná hodnota	Stínová cena	Omezující podmínka Pravá strana	Povolený nárůst	Povolený pokles
D2	střihací stroj skalární součin	480	5	480	240	360
D3	šicí stroj skalární součin	480	2,5	480	1440	180

Obr. 4 Citlivostní zpráva 1

Jak již bylo uvedeno, budeme se v tomto článku věnovat citlivostní zprávě. Tato zpráva obsahuje řadu dalších zajímavých informací, kterých může učitel využít při diskusi o řešení úloh lineárního programování se studenty. Předpokladem této činnosti je správně pochopit některé pojmy (intervaly stability pro cenové koeficienty, redukované ceny, stínové ceny, intervaly stability pro pravé strany podmínek) a dokázat je interpretovat pro konkrétní podmínky dané úlohy. Podrobné vysvětlení uvedených pojmů lze nalézt například v publikaci [3]. V následujících odstavcích se budeme věnovat rozboru těchto pojmů pro náš konkrétní případ výroby sešitů.

2.1 Snížené náklady a intervaly stability pro cenové koeficienty

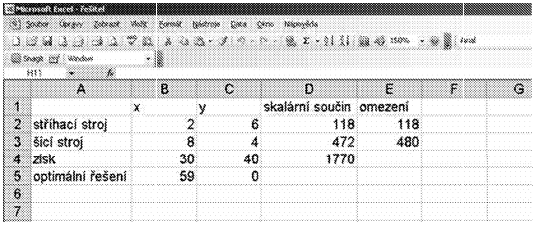
Všimněme si nejprve první části tabulky, viz obr. 4, nadepsané „Měněné buňky“. „Konečná hodnota“ zde zřejmě představuje vypočtené optimální řešení $x = 24$, $y = 72$ a „Cílový koeficient“ (správně by mělo být „Cenový koeficient“) udává zisk z prodeje jednoho sešitu typu A, resp. typu B, tj. 50 Kč, resp. 40 Kč. Ve sloupci „Snížené náklady“ lze přečíst tzv. redukované ceny, které udávají maximální možnou změnu cenových koeficientů, která ještě neovlivní celkový zisk. Vzhledem k tomu, že v našem případě jsou obě hodnoty v tomto sloupci nulové, znamená to, že jakákoliv změna cenových koeficientů má za následek změnu celkového zisku. Poslední dva sloupce s nadpisem „Povolený nárůst“ a „Povolený pokles“ udávají tzv. intervaly stability pro cenové koeficienty. Pokud budou např. hodnoty prvního cenového koeficientu 30 měněny v intervalu od $30 - 16,66666667$ do $30 + 50$, optimální řešení $x = 24$, $y = 72$ se nezmění (změní se pouze

celkový zisk z). Poznamenejme, že ve skutečnosti by Excel bral za krajní body intervalu stability přibližné hodnoty 13,3334 a 79,9999. Snadno se můžeme přesvědčit, že pro hodnotu cenového koeficientu 13,3333 se optimální řešení skokem mění na $x = 0$, $y = 80$ a pro hodnotu 80 na $x = 60$, $y = 0$. Pokud by tedy zisk z prodeje jednoho sešitu typu A klesl pod spodní hranici intervalu stability (13,3334 Kč), je již výhodnější vyrábět pouze sešity typu B, kterých se vyrobí 80 kusů. Vzroste-li naopak tento zisk nad horní hranici intervalu stability (79,9999 Kč), je výhodnější vyrábět pouze sešity typu A, kterých se vyrobí 60 kusů.

2.2 Stínové ceny a intervaly stability pro omezující podmínky

Ve druhé části tabulky nadepsané „Omezující podmínky“ jsou nejprve uvedeny „Konečné hodnoty“, které vyjadřují reálný čas, po který byly oba stroje během směny v provozu (každý 480 minut). Porovnáme-li tyto hodnoty s hodnotami ve třetím sloupci („Omezující podmínka – Pravá strana“), vidíme, že strojový čas byl v obou případech využit stoprocentně. Tuto informaci lze samozřejmě vyčíst i z výstupu na obr. 2 (buňky D2, D3). Hodnoty ve sloupci s nadpisem „Stínová cena“ udávají nárůst celkového zisku v korunách při navýšení pravé strany příslušné omezující podmínky o jednotku (v našem případě o 1 minutu). Pokud by tedy např. stříhací stroj pracoval o 1 minutu déle, pak by při zachování všech ostatních podmínek vzrostl zisk o 5 Kč. Prodloužením pracovní doby šicího stroje o 1 minutu by zisk vzrostl o 2,50 Kč. Poznamenejme, že hodnoty stínových cen 5 a 2,5 rovněž určují koeficienty modelu tzv. duální úlohy lineárního programování (viz např. [3]), ale tato problematika již přesahuje rámec možného středoškolského učiva.

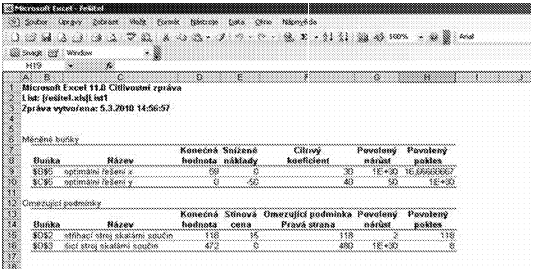
Poslední dva sloupce udávají tzv. intervaly stability pro pravé strany omezujících podmínek. Například povolený nárůst pro pravou stranu první omezující podmínky je 240 a povolený pokles 360, tj. první stroj, který provádí stříhání papíru, může pracovat buď o 240 minut déle nebo o 360 minut méně, aniž by došlo ke změně optimálního rozvržení výroby. Znamená to, že vzroste-li hodnota pravé strany první nerovnice modelu o více než 240 nebo klesne-li o více než 360, přestává být výroba jednoho z typů sešitu výhodná. Konkrétně, pro hodnotu pravé strany první podmínky 118, dostáváme optimální řešení $x = 59$, $y = 0$ (obr. 5), tj. pokud pracovní využití prvního stroje nepřevýší hodnotu 118 minut, je optimální vyrábět 59 kusů sešitu A se ziskem 1770 Kč. Vidíme rovněž, že čas druhého stroje, tj. šicího, nebude v tomto případě zcela využit (buňka D3).



	A	B	C	D	E	F	G
1		x	y	skalární součin	omezení		
2	střihací stroj	2	6	118	118		
3	šicí stroj	8	4	472	480		
4	zisk	30	40	1770			
5	optimální řešení	59	0				
6							
7							

Obr. 5 Výstup MS Excelu po výpočtu optimálního řešení *Řešitelem 2*

Z citlivostní zprávy (obr. 6) lze také vyčíst, že snížené náklady pro y nyní činí -50. To lze interpretovat tak, že zvýšením zisku z prodeje jednoho sešitu typu B o jakoukoliv hodnotu nepřesahující 50 Kč se celkový zisk nezmění. Pokud tedy bude zisk ze sešitu B menší než 90 Kč, stále je výhodnější vyrábět pouze sešit typu A.



Microsoft Excel 11.0 Citlivostní zpráva
List: [sesitel.xls]List1
Zpráva vytvořena: 5.3.2010 14:56:57

Měněné buňky

Buňka	Název	Konečná hodnota	Snížené náklady	Citlivý koeficient	Povolený nárůst	Povolený pokles
D5	optimální řešení x	59	0	30	1E+30	10,00000000
C5	optimální řešení y	0	-50	40	50	1E+30

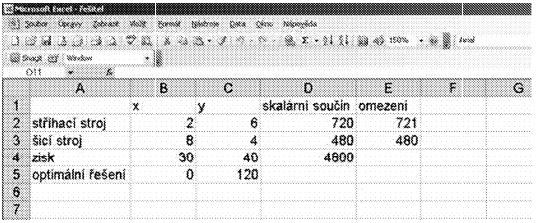
Omezující podmínky

Buňka	Název	Konečná hodnota	Stínová cena	Omezující podmínka Pravá strana	Povolený nárůst	Povolený pokles
D2	střihací stroj skalární součin	118	15	118	2	118
D3	šicí stroj skalární součin	472	0	480	1E+30	8

Obr. 6 Citlivostní zpráva 2

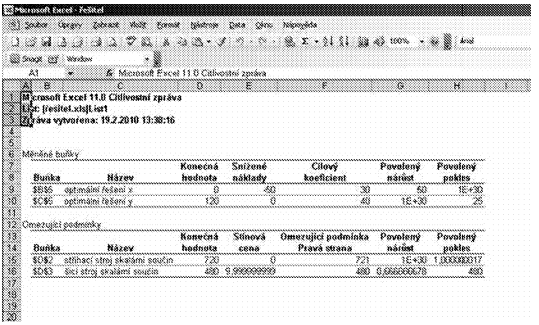
Bude-li hodnota pravé strany první podmínky 721, dostaneme optimální řešení $x = 0$ a $y = 120$ (obr. 7), tj. pokud pracovní využití prvního stroje neklesne pod hodnotu 721 minut, je optimální vyrábět 120 kusů sešitu B se ziskem 4800 Kč. Vidíme rovněž, že čas druhého stroje, tj. šicího, bude v tomto případě zcela využit (buňka D3).

V tomto případě lze vyčíst z citlivostní zprávy (obr. 8), že snížené náklady pro x činí -50. To lze interpretovat tak, že zvýšením zisku z prodeje jednoho sešitu typu A o jakoukoliv hodnotu nepřesahující 50 Kč se celkový zisk nezmění. V případě, že zisk ze sešitu A bude menší než 80 Kč, je stále výhodnější vyrábět pouze sešit typu B.



	A	B	C	D	E	F	G
1		x	y	skalární součin	omezení		
2	střihací stroj	2	6	720	721		
3	šicí stroj	8	4	480	480		
4	zisk	30	40	4800			
5	optimální řešení	0	120				
6							
7							

Obr. 7 Výstup MS Excelu po výpočtu optimálního řešení *Řešitelem 3*



Obr. 8 Citlivostní zpráva 3

Podobnou diskusi lze provést i pro druhou podmínku.

Všechny uvedené vztahy lze studentům demonstrovat postupným měněním příslušných buněk excelovského sešitu a následným výpočtem nového optimálního řešení *Řešitelem*, příp. vygenerováním nové citlivostní zprávy. Poznamenejme, že v některých případech můžeme dostat neceločíselné optimální řešení, které nemá pro naši úlohu reálný smysl. Tuto nepříjemnost lze odstranit doplněním podmínek celočíselnosti pro měněné buňky B5 a C5 v dialogovém okně „Parametry Řešitele“. Po přidání těchto podmínek již Řešitel určí celočíselné optimální řešení, nevygeneruje však citlivostní zprávu.

3. Závěr

Uvedené ukázky naznačují, že *Řešitel* může být opravdu užitečným pomocníkem usnadňujícím a zpestřujícím výuku matematiky a dalších středoškolských předmětů. Jakým způsobem se jej podaří ve výuce využít již záleží na učitelově kreativitě a invenci.

Literatura

[1] *Pražák, P.– Pražáková, B.*: Základní použití Řešitele v Excelu, Matematika-fyzika-informatika.18, 433–441, 2009.

[2] *Brož, M.*: Microsoft Excel pro manažery a ekonomy, Computer Press, a. s, Brno 2006.

[3] *Jablonský, J.*: Operační výzkum, kvantitativní modely pro ekonomické rozhodování, Professional Publishing, Praha 2002.

ZPRÁVY

Ústřední kolo 59. ročníku MO

Ústřední kolo 59. ročníku Matematické olympiády v kategorii A se uskutečnilo v termínu 21. – 24. 3. 2010 v Chebu. Organizátorem III. (závěrečného) kola soutěže byla v letošním roce Krajská komise MO v Karlovarském kraji. Záštitu nad finálovou částí MO převzali *PaedDr. Josef Novotný*, hejtman Karlovarského kraje, *MUDr. Jan Svoboda*, starosta města Chebu a *Mons. František Radkovský*, biskup plzeňský, kteří se zúčastnili slavnostního zahájení ústředního kola. To se uskutečnilo v neděli 21. března 2010 v prostorách Fakulty ekonomické Západočeské Univerzity v Chebu. Kromě soutěžících, členů ÚK MO a garantů se zahájení soutěže zúčastnili rovněž pozvaní hosté, mezi nimiž byli především zástupci společenského života v Chebu a v Karlovarském kraji a dále zástupci sponzorů (skupina ČEZ, ECOVIS Corporate Finance s. r. o.). Soutěžící a členové Ústřední komise MO (ÚK MO) byli ubytováni v Domově mládeže Střední zdravotnické školy v Chebu. Vlastní soutěž se přitom po oba soutěžní dny konala v učebnách Gymnázia Cheb.

Na základě jednotné koordinace úloh krajského (II.) kola v kategorii A pozvala ÚK MO k účasti ve III. kole 51 nejlepších řešitelů z celé republiky (jeden z nich se však předem omluvil). Svého zástupce v ústředním kole neměl tentokrát pouze Liberecký kraj. Soutěžními dny byly 22. a 23. březen 2010.

Na řešení obou trojic soutěžních úloh měli soutěžící již tradičně vyhrazeny vždy 4,5 hodiny čistého času a za každou úlohu mohli přitom získat maximálně 7 bodů.

Chebští organizátoři připravili pro soutěžící a pro členy ÚK MO atraktivní doprovodný program. Odpoledne po prvním soutěžním dnu bylo vyhrazeno zájezdu do přírodní rezervace SOOS poblíž Františkových Lázní a prohlídce historického centra Chebu. Následující den odpoledne byla pro zájemce zajištěna prohlídka zámku Kynžvart spojená s návštěvou nedalekých Mariánských Lázní.

Slavnostní vyhlášení výsledků a předání cen nejlepším soutěžícím se uskutečnilo ve středu 24. března 2010 dopoledne v aule FEK ZČU v Chebu za přítomnosti představitelů města Plzeň a zástupců ZČU v Plzni. Předseda ÚK MO *doc. Jaromír Šimša* poděkoval ve svém závěrečném projevu všem, kteří se zasloužili o zdárný průběh ústředního kola MO v kategorii A, především pak předsedovi Krajské komise MO v Karlovarském kraji – *Mgr. Josefu Hazimu* – a krátce informoval všechny přítomné o ústředním kole v jubilejním, 60. ročníku MO, které se uskuteční v terminu 27. – 30. 3. 2011 v Brně.