Vítejte v prostředí Mathematica

Srdečně Vás vítá kantor Pavel Stříž!

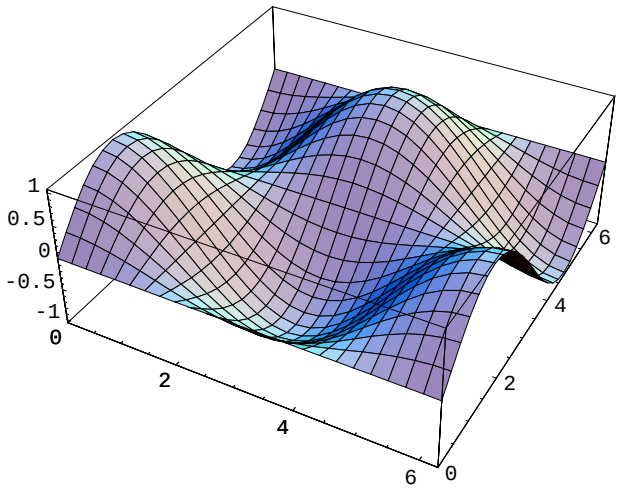
První ukázky

```
In[3]:= 8 / 9
```

$$\text{Out[3]}= \frac{8}{9}$$

```
In[4]:= 8 / 9 // N

Out[4]= 0.888889

In[5]:= 12!

Out[5]= 479001600

In[6]:= 123!

Out[6]= 1214630436702532967576624324188129585545421708848338231532891816182923589236216766
        883115696061264020217073583522129404778259109157041165147218602951990626164673073
        390741981495296000000000000000000000000000
```

```
In[7]:= Plot3D[Cos[x] Sin[y], {x, 0, 2 Pi}, {y, 0, 2 Pi}];
```



Optimalizujeme: Nakupujeme ovoce

■ Zadání základní úlohy

Mám 100 Kč. Jedno jablko stojí 8 Kč (formálně 8 Kč/ks), jedna hruška stojí 12 Kč. Kolik kusů jablek a hrušek máme koupit, abychom měli co největší počet ovoce?

■ Řešení úlohy

Pro počet koupených jablek si zavedeme proměnnou x, pro počet zvažovaných hrušek proměnnou y. Obě proměnné mohou být větší nebo rovno nule, záporné hodnoty (ve smyslu my prodáváme) nemají pro nás význam. Tedy formálně $x \geq 0,\ y \geq 0$.
Maximální počet koupených jen jablek či jen hrušek za 100 bychom získali snadno, ale teď od této úvahy upustíme.
Za všechny jablka obecně utratíme $8 * x$, za hrušky pak $12 * y$. Celkem utratíme podle nákupu $8\,x + 12\,y$.
Nemůžeme však utratit více než naši stokorunu, tedy $8\,x + 12\,y \leq 100$. Levá i pravá strana nerovnice je v korunách.
Formálně chceme maximalizovat počet kusů jablek a hrušek, tedy $x + y \to \max$.
Více zadání neomezuje. Přejděme k matematickým propočtům. Optimalizujeme a užijeme funkce `Maximize` nebo `NMaximize`. Parametry a syntaxi lze vyčíst v `Help Browseru`. Funkci `N` použijeme k rozumnému zobrazení výsledku.

```
In[8]:= Nakup = Maximize[x + y, x ≥ 0 && y ≥ 0 && 8 x + 12 y ≤ 100, {x, y}]
        N[Nakup]
```

$$\text{Out[8]= } \left\{\frac{25}{2}, \left\{x \to \frac{25}{2}, y \to 0\right\}\right\}$$

```
Out[9]= {12.5, {x → 12.5, y → 0.}}
```

Nesmíme zapomenout, že nelze koupit necelý kus ovoce (to by nám pan nebo paní prodavačka dala co proto), proto připustíme jen celá čísla pomocí $\{x, y\} \in$ Integers.

```
In[10]:= Nakup = Maximize[x + y, x ≥ 0 && y ≥ 0 && 8 x + 12 y ≤ 100 && {x, y} ∈ Integers, {x, y}]
         N[Nakup]
```

```
Out[10]= {12, {x → 12, y → 0}}
```

```
Out[11]= {12., {x → 12., y → 0.}}
```

Stejný výsledek zapsaný pomocí funkce `NMaximize`. Do seznamu si uložíme x a y. Vypočítáme si, kolik nám zůstane.

```
In[12]:= Nakup =
          {x, y} /. NMaximize[{x + y, x ≥ 0, y ≥ 0, 8 x + 12 y ≤ 100, {x, y} ∈ Integers}, {x, y}][[2]]
```

```
Out[12]= {11, 1}
```

```
In[13]:= 100 - 8 * Nakup[[1]] - 12 * Nakup[[2]]
```

```
Out[13]= 0
```

■ Interpretace úlohy

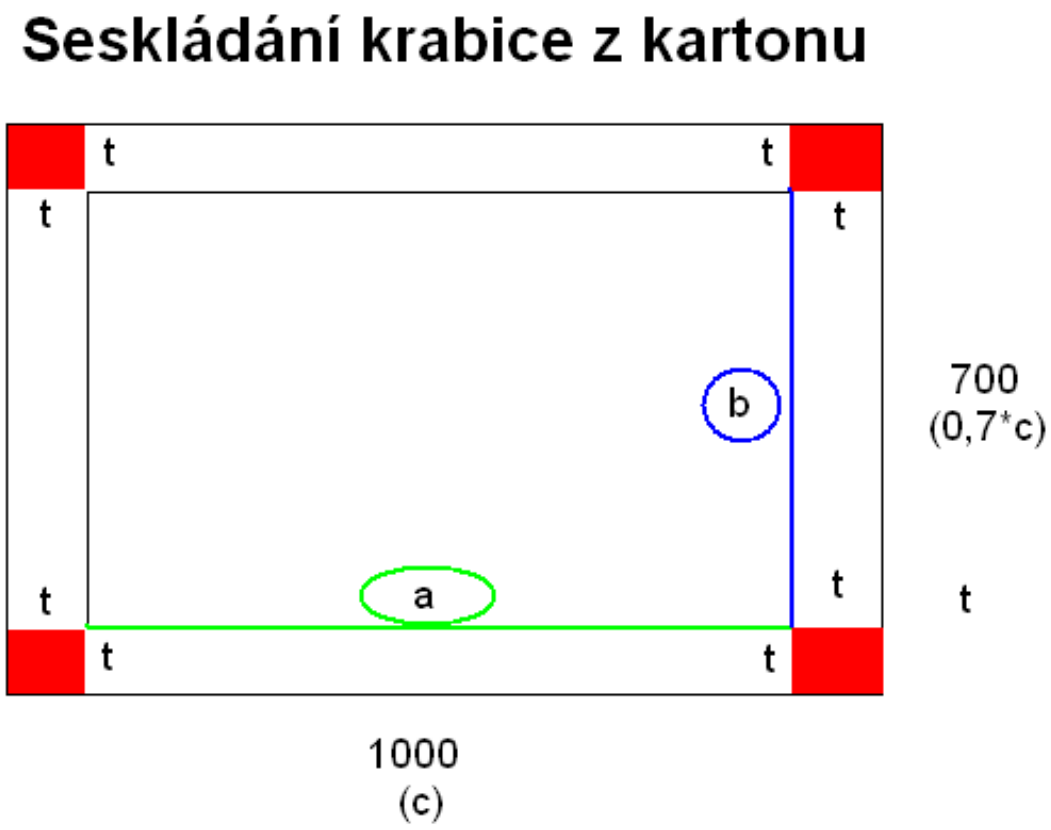
Pokud chceme mít co největší počet kusů ovoce a máme 100 Kč, tak při všech omezeních koupíme 11 kusů jablíček a 1 kus hrušky. Získáme tak 12 kusů ovoce. Pomocí odečtení $100 - 8\,x + 12\,y$ zjistíme, že nám nezůstane ze stokoruny nic.

Optimalizujeme: Objem kvádru bez poklopu

■ Zadání příkladu

Mějme ještě jeden příklad. Mějme karton o velikosti 1000 krát 700 milimetrů (obecně pak c, tedy $0{,}7 * c$). Máme sestavit kvádr o maximálním objemu bez poklopu.

```
In[14]:= Show[Import["http://exp.uis.fame.utb.cz/vyuka/pzd/zdroj-kvadr.png"]]
```




```
Out[14]= - Graphics -
```

■ Řešení příkladu

Zadání je znázorněno na obrázku, kdy červené čtverce vyřežeme. Krabice, nebo-li kvádr, se seskládá sklopením všech čtyř stěn. Délky t jsou všude stejné, různé nemají smysl. Objem otevřené krabice tedy pak získáme jako $V = a * b * t$. Pokud se chvíli zamyslíme, zjistíme, že nemůže nastat $a > 1000$, $b > 700$. Další věci si určitě všimnete, a to, že nemůže nastat $t > 350$, poněvadž by nebylo jak krabici složit po kratší hraně. Určitě musí platit že a, b, $t \geq 0$.
Omezení můžeme upřesnit, a to $a + 2 * t \leq 1000$ a $b + 2 * t \leq 700$. Znaménkem menší umožňujeme to, že nemusí být celý karton spotřebován, což tušíte nejspíše nenastane. Ale z hlediska matematických výpočtů, těch technických v pozadí, je $\geq$ či $\leq$ téměř vždy výhodnější než rovnítko.
Všechny tyto omezení vložíme do funkce `Maximize` nebo `NMaximize`, a optimalizujeme přes tři parametry a, b a t. Výsledky si necháme upravit přes funkci `N`.

```
In[15]:= Vysledek = Maximize[a * b * t, a + 2 * t ≤ 1000 && b + 2 * t ≤ 700 &&
           t ≤ 350 && a ≥ 0 && b ≥ 0 && t ≥ 0 && a <= 1000 && b <= 700 && t ≤ 350, {a, b, t}]
         N[Vysledek, 10]
         N[Vysledek]
```

$$\text{Out[15]= } \left\{-\frac{1000000}{27}\left(-442-79\sqrt{79}\right),\ \left\{a \to -\frac{200\left(-442-79\sqrt{79}\right)}{3\left(17-\sqrt{79}\right)\left(4+\sqrt{79}\right)},\ b \to \frac{100}{3}\left(4+\sqrt{79}\right),\ t \to \frac{50}{3}\left(17-\sqrt{79}\right)\right\}\right\}$$

```
Out[16]= {4.237656885 × 10^7, {a → 729.6064806, b → 429.6064806, t → 135.1967597}}

Out[17]= {4.23766 × 10^7, {a → 729.606, b → 429.606, t → 135.197}}
```

Při úvahách můžeme délky nahradit parametrem c, který si v úvodu vyčistíme a definujeme. Symbolicky se při optimalizaci častokrát nepracuje. Numerické výsledky jsou stejné.

```
In[18]:= Clear[c]
         c := 1000
         Result = Maximize[{a * b * t, a + 2 * t ≤ c && b + 2 * t ≤ 0.7 * c &&
             t ≤ 0.35 * c && a ≥ 0 && b ≥ 0 && t ≥ 0 && a < c && b < c && t < c}, {a, b, t}]

Out[20]= {4.23766 × 10^7, {a → 729.606, b → 429.606, t → 135.197}}

In[21]:= N[Result, 7]

Out[21]= {4.23766 × 10^7, {a → 729.606, b → 429.606, t → 135.197}}
```

Funkci spočteme přes `NMaximize`, tedy ne symbolicky, ale přímo numericky. Více o rozdílech v Helpu či manuálech.

```
In[22]:= Clear[c]
         c := 1000
         Result = NMaximize[{a * b * t, a + 2 * t ≤ c && b + 2 * t ≤ 0.7 * c &&
             t ≤ 0.35 * c && a ≥ 0 && b ≥ 0 && t ≥ 0 && a < c && b < c && t < c}, {a, b, t}]

Out[24]= {4.23766 × 10^7, {a → 729.606, b → 429.606, t → 135.197}}
```

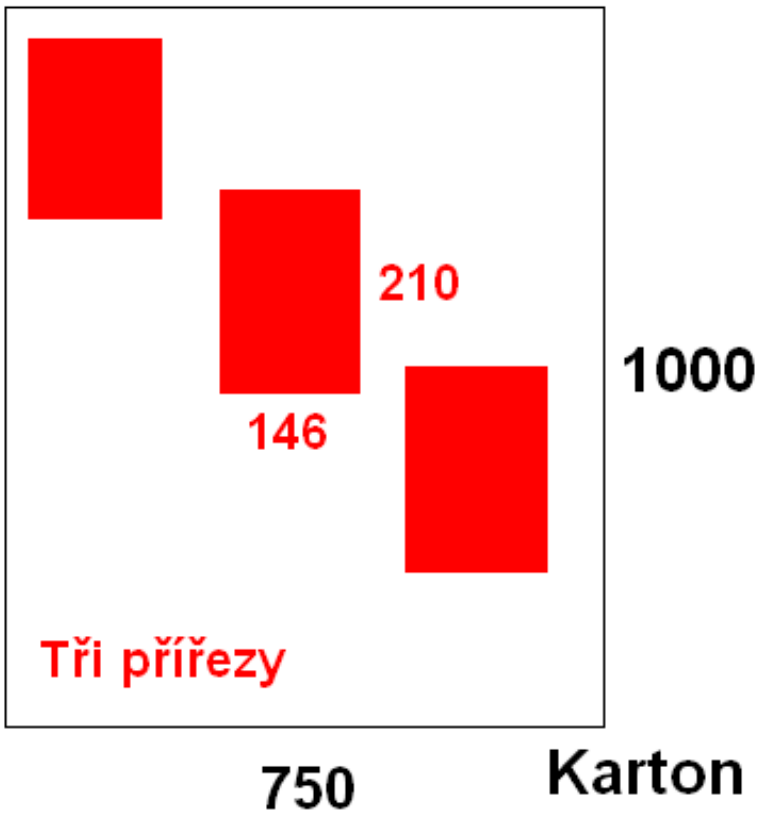
■ Interpretace příkladu

Pokud máme karton 1000 krát 700 milimetrů, lze dosáhnout maximálního objemu 42,3766 litru (decimetrů krychlových). Dosáhneme toho při rozměrech 729,606 krát 429,606 krát 135,197 milimetrů. Žádná jiná kombinace rozměrů nám nepřinese větší objem kvádru. Hezký příklad, ne?

Bonusový příklad ke zkoušce (bez řešení, jen s tipy)

■ Zadání a vysvětlující komentáře

```
In[25]:= Show[Import["http://exp.uis.fame.utb.cz/vyuka/pzd/zdroj-zaver.PNG"]]
```




```
Out[25]= - Graphics -
```

Tohle je praktický příklad, který kantor řešil v prosinci roku 2007. Mějte knihařský karton 1000 krát 750 milimetrů (a tloušťce 2,0 mm, ale to není pro náš příklad důležitý údaj). Z něj řežete na pákové řezačce přířezy, které tvoří základ vázaných knih. Řezat můžete svisle nebo vodorovně po celé délce, tedy vždy rovnoběžně s jednou z hran kartonu (nemáte potřebné zarážky, abyste nastavovali přesný úhel řezu, tedy můžete jen otočit kartonem o 0, 90, 180, ... stupňů).
Kolik přířezů 146 krát 210 (resp. 210 krát 146, to je po nařezání jedno a to samé) můžete z jednoho kartonu maximálně získat? Nepočítejte úbytky způsobené vnitřními řezy (ve skutečnosti je to přítlakem zdeformovaný asi milimetr až dva, které je lépe ze strany odříznout).

Řešením se můžete pochlubit u zkoušky za bonusových 10 procent ze 100+10 možných (což je přibližně jeden známkový stupeň dle ECTS). Řešení můžete mít na papíře, v Microsoft Excelu, MATLABu, MATHEMATICA, či jiném programu. Maximalizujeme jistou veličinu, tedy se jedná o oblast optimalizace.
Logická úvaha. Plocha kartonu je 1000 krát 750, tedy 750000 mm čtverečních. Plocha jednoho přířezu je 146 krát 210, tedy 30660 mm čtverečních. Určitě nepůjde získat více jak 750000/30660 přířezů, tedy konkrétně 24,46. Tedy 25 přířezů nikdy nezískáte, není dostatek kartonu. Dle množství materiálu by snad šlo získat 24 přířezů. Kolik je to však ve skutečnosti?
Na obrázku jsem dokázal nařezat tři přířezy, půjde to však lépe, ne? Zkuste!

Prozatím se loučí kantor
Pavel Stříž