Krájení pizzy
(Úlohy z MO kategorie P, 45. část)

PAVEL TÖPFER

Matematicko-fyzikální fakulta UK, Praha

V dnešním dílu seriálu o zajímavých programátorských problémech z Matematické olympiády kategorie P se seznámíme s jednou praktickou úlohou z domácího kola 56. ročníku MO (školní rok 2006/07). Jedná se o úlohu poměrně snadnou a v olympiádě poněkud neobvyklou. Je to úloha optimalizační, k jejímuž vyřešení ale nepotřebujeme žádné zvláštní znalosti algoritmů. Úplně nám postačí jednoduchá logická úvaha a „hladový" přístup k řešení problému. Ve druhé části článku se pak seznámíme s trochu obtížnější variantou úlohy, která bude na první pohled vypadat odlišně, ale k řešení použijeme prakticky stejný postup.

Nejprve se jako obvykle seznámíme s přesným zadáním úlohy. Některé jeho formulace jsme pro potřeby článku trochu upravili, aniž bychom tím ovšem změnili smysl úlohy.

* * * * * * * * * * * *

Marco se rozhodl, že zužitkuje své kulinářské i cyklistické dovednosti a založí si firmu pro výrobu a rozvoz pizzy. Firmu plánuje provozovat tak, že vždy nejdříve bude shromažďovat objednávky a když jich bude dostatek, tak pizzy upeče, sedne na kolo a rozveze je zákazníkům. Protože Marco je lepší cyklista než kuchař, zatím ve své nabídce plánuje pouze jeden druh pizzy. Aby se ale odlišil od konkurence, přijímá objednávky i na šestinové části pizzy. Lze si u něj objednat například 1/6, 4/6 nebo 15/6 pizzy. Navíc jako speciální službu zákazníkům chce Marco dodat každému zákazníkovi pizzy co nejméně rozřezané, aby si zákazník sám mohl rozhodnout, jak si dále pizzu rozdělí. Proto například 4/6 pizzy chce dodat jako jeden kus příslušné velikosti a 15/6 chce dodat jako dvě celé pizzy a k nim jednu polovinu pizzy. Marco chce vždy dodat co nejvíce pizz vcelku, takže uspokojení této objednávky například třemi kusy velikosti 5/6 nepřipadá v úvahu.

Když Marco všude rozdal letáky propagující jeho novou firmu, uvědomil si, že díky jeho speciální službě zákazníkům není jednoduché zkombinovat

objednávky tak, aby mu moc kusů pizzy nezbylo. Obrátil se proto na vás, abyste mu napsali program, který by mu s problémem pomohl. Pro začátek bude stačit program, který na vstupu dostane seznam objednávek a na výstup vypíše, kolik pizz má Marco napéct.

Formát vstupu

Na vstupu se nachází n kladných celých čísel, z nichž každé popisuje jednu objednávku a udává počet objednaných šestin pizzy.

Formát výstupu

Program vypíše jedno celé číslo p, které značí nejmenší možný počet pizz, které je třeba upéct, aby šlo splnit všechny objednávky a byla dodržena výše uvedená speciální služba zákazníkům.

Příklad 1

Vstup:	Výstup:
2 2 3	2

Vysvětlení: Z jedné pizzy lze například uříznout dva kusy o velikosti 2/6 a z druhé pizzy se uřízne kus velký 3/6.

Příklad 2

Vstup:	Výstup:
4 5 3	3

Vysvětlení: Kvůli požadavku na dodání co nejméně rozřezaných kousků pizzy je třeba pro každou objednávku upéct celou pizzu.

* * * * * * * * * * * *

Hned při čtení vstupních dat můžeme snadno určit, kolik budeme celkově zákazníkům doručovat celých pizz a také potřebné počty kusů jednotlivých velikostí 1/6, 2/6, 3/6, 4/6 a 5/6. Z těchto údajů pak spočítáme výsledný počet pizz p.

Na doručovaných celých pizzách není co řešit, ty do počtu p samozřejmě musíme započítat. Dílčí kusy pizzy budeme nyní zpracovávat podle velikosti od největších k nejmenším. Pro každý kus velký 5/6 potřebujeme zjevně upéct jednu celou pizzu. Po každém kusu velkém 5/6 zbude jeden kousek velký 1/6 a tyto kousky použijeme na pokrytí objednávek na kousky této velikosti. Pokud je těchto zbytků velkých 1/6 více, než kolik vyžadují objednávky, nezbývá nám než přebývající kousky 1/6 vyhodit.

Pro objednané kusy velké 4/6 je situace podobná jako pro kusy velké 5/6. Pro každý z nich opět potřebujeme upéct jednu další celou pizzu a zbytky velikosti 2/6 použijeme na pokrytí objednávek na kusy velké 2/6. Je-li takových objednávek málo, tak některé zbytky velké 2/6 ještě rozřežeme a použijeme na pokrytí objednávek na kousky velikosti 1/6, pokud nějaké zbývají. Zjevně je výhodnější přednostně uspokojit objednávky na velikost 2/6 a až pak na velikost 1/6, protože kousek veliký 1/6 můžeme z libovolně velkého kusu odříznout vždy.

Nyní se dostáváme k objednávkám na kusy velikosti $3/6 = 1/2$. Objednávky na části pizzy velikosti 1/2 vyřešíme tak, že vždy jednu pizzu rozdělíme na dvě poloviny. Pokud je počet takových objednávek lichý, zbude nám jedna polovina pizzy. Tu rozdělíme na jeden kousek velikosti 2/6 a jeden kousek velikosti 1/6, s nimiž naložíme stejně, jak jsme právě popsali.

Jestliže nám nyní ještě zbyly nějaké nevyřízené objednávky na kusy velké 2/6 (což je 1/3), budeme péct další pizzy a dělit je na třetiny. Případný zbytek po pokrytí všech objednávek na 2/6 pak použijeme na zbývající objednávky kousků velikosti 1/6. Pokud nakonec zůstaly ještě nevyřízené objednávky na kousky velikosti 1/6, upečeme ještě další pizzy na jejich pokrytí.

Uvedený postup můžeme ještě značně zjednodušit. Nejprve napočítáme pizzy potřebné pro správné vyřízení objednávek na kusy velikosti 5/6 a 4/6 a jejich zbytky použijeme na vyřízení objednávek 1/6 a 2/6, jak je popsáno výše. Dále ale už stačí sečíst velikost všech objednaných kusů velikosti 3/6 a dosud nevyřízených objednávek na kusy velikosti 2/6 a 1/6. Tím zjistíme, kolik dalších pizz musíme ještě upéct pro vyřízení všech těchto objednávek. Výslednou hodnotu p tedy zvýšíme o horní celou část získaného součtu.

Popsané zjednodušení si můžeme dovolit proto, že zpracování všech objednávek na kousky 3/6 a zbývajících objednávek na kousky 2/6 a 1/6 je téměř bezeztrátové. Pokud je objednávek na kusy 3/6 sudý počet, pak všechny další pizzy plně využijeme na kousky 2/6 a 1/6 bez ohledu na to, kolik je objednáno kterých, až z úplně poslední pizzy nám může něco zbýt (nejvýše 5/6). Pokud je objednávek na kusy 3/6 lichý a zbývá alespoň jedna nevyřízená objednávka 1/6, odkrojíme tuto objednanou jednu 1/6 ze zbývající poloviny pizzy a tím převedeme situaci na předchozí případ. Maličko složitější situace nastane jedině v případě, že počet objednávek na kusy velikosti 3/6 je lichý a již nezbývá žádná nevyřízená objednávka na kousek 1/6. V takovém případě nám od poslední objednané poloviny

pizzy zůstane lichá polovina, z níž ukrojíme kousek velikosti 2/6 a pro zbývající 1/6 již nemáme využití. Zbývající objednávky kousků velikosti 2/6 budeme již vyřizovat z dalších pizz. I v tomto případě ovšem funguje náš zjednodušený postup výpočtu a také i v tomto případě bude platit, že nám nakonec zbude nejvýše 5/6 pizzy.

Algoritmus má lineární časovou složitost vzhledem k počtu objednávek n, neboť každou objednávku musíme zpracovat a určit počty objednaných kusů jednotlivých velikostí. Další výpočet už má časovou složitost konstantní, nezávislou na n. Prostorová složitost algoritmu je rovněž konstantní, při výpočtu nepotřebujeme žádnou datovou strukturu velikosti n, vystačíme jen s konstantním počtem pracovních proměnných.

Ukázkový program v Pythonu je jen přímým přepsáním výše uvedených úvah o dělení pizzy a je překvapivě krátký:

```
vstup = [int(_) for _ in input().split()]

obj = [0] * 6              # počty objednávek
p = 0                      # výsledný počet pizz
for x in vstup:
    p += x // 6            # počet celých pizz v objednávkách
    obj[x % 6] += 1        # počty objednaných kousků pizzy

p += obj[5] + obj[4]
obj[1] -= obj[5]
obj[2] -= obj[4]
if obj[2] < 0:
    obj[1] += 2 * obj[2]
    obj[2] = 0
if obj[1] < 0:
    obj[1] = 0

z = 3 * obj[3] + 2 * obj[2] + obj[1]
p += z // 6
if z % 6 > 0:
    p += 1

print(p)
```

Ve druhé části článku si ukážeme ještě jednou úlohu, která na první pohled sice vypadá odlišně, ale při jejím řešení použijeme naprosto shodný postup. Zatímco krájení pizz na kousky je vlastně dělením kruhů na kru-

hové výseče a probíhá tedy v dvourozměrné rovině, nyní budeme skládat krychlové krabice do přepravních palet a tím se přesuneme do trojrozměrného prostoru. Správné řešení tak od nás bude vyžadovat i trochu prostorové představivosti. Začneme jako obvykle zadáním problému.

* * * * * * * * * * * *

Továrna expeduje své výrobky v krabicích tvaru krychle. Používá šest velikostí krabic o hraně délky 20 cm, 40 cm, 60 cm, 80 cm, 100 cm a 120 cm. Zabalené výrobky se odvážejí v přepravních paletách, které mají tvar krychle o hraně 120 cm. Vstupem programu je šest nezáporných celých čísel, která představují počty jednotlivých druhů krabic připravených na odvoz, a to v pořadí od nejmenších krabic po největší. Určete, jaký minimální počet palet je třeba použít k odvozu všech těchto krabic s výrobky. Krabice se při ukládání do palet nesmějí nijak deformovat. Výsledkem výpočtu bude jedno celé číslo představující počet potřebných palet.

Příklad

Vstup:	Výstup:
3 0 0 2 0 1	3

Vysvětlení: Chceme odvézt 3 krabice o hraně 20 cm, 2 krabice o hraně 80 cm a 1 krabici o hraně 120 cm, žádné krabice zbývajících tří velikostí se neodvážejí.

K odvozu těchto krabic jsou zapotřebí tři palety. Největší krabice zcela zaplní samostatnou paletu, všechny zbývající krabice by se z hlediska svého objemu vešly do jedné další palety, ale protože se nesmějí deformovat, musíme použít ještě dvě palety.

* * * * * * * * * * * *

Řešení tohoto problému do značné míry kopíruje postup, který jsme si popsali při rozboru předchozí úlohy s krájením pizzy. Krabice budeme opět zpracovávat v pořadí od největších k nejmenším, počty těchto přepravovaných krabic označíme postupně p_6, p_5, p_4, p_3, p_2 a p_1. V proměnné p budeme počítat minimální nezbytné množství přepravních palet.

Pro každou krabici velikosti 120 cm jistě potřebujeme samostatnou paletu, kterou tato krabice zcela zaplní. Pro každou krabici velikosti 100 cm potřebujeme také novou paletu. Když do ní krabici vložíme, zůstane při jejích třech sousedních stěnách volný prostor, který můžeme využít jedině

na nejmenší krabice velikosti 20 cm. Těchto malých krabice se tam vejde 91. U jedné stěny totiž bude $6 \times 6 = 36$ krabic, kvůli společné hraně u druhé stěny zbývá místo na $6 \times 5 = 30$ krabic, konečně u třetí stěny vzhledem ke společným hranám s oběma předchozími stěnami umístíme ještě dalších $5 \times 5 = 25$ krabic. Tuto „volnou kapacitu“ v existujících paletách využitelnou pro krabice velikosti 20 cm si označíme x_1.

Podobné je to s krabicemi velikosti 80 cm. Pro každou z nich potřebujeme založit novou paletu a v ní nám zbude volný prostor pro 19 krabic velikosti 40 cm. Žádnou větší krabici tam umístit nemůžeme. Naopak místo kterékoliv krabice velikosti 40 cm můžeme dát 8 malých krabic velikosti 20 cm, což možná časem uděláme v závislosti na konkrétních vstupních datech. Přednostně se ale budeme snažit umístit do existujících palet větší krabice 40 cm. Když už nebudeme žádné mít, nezaplněný prostor vždy můžeme využít pro případné zbývající krabice 20 cm. Zavedeme si označení x_2 pro volnou kapacitu v existujících paletách využitelnou pro krabice velikosti 40 cm. Zatím tedy máme $p_6 + p_5 + p_4$ nezbytných palet, do nichž jsme umístili všechny krabice velikosti 120 cm, 100 cm a 80 cm. V těchto paletách nám zůstal volný prostor pro x_2 krabic velikosti 40 cm a ještě pro dalších x_1 krabic velikosti 20 cm.

Nejvíc práce budeme mít s krabicemi velikosti 60 cm. Ty můžeme krásně skládat do palet vždy po osmi, osm těchto krabic paletu zcela zaplní. Pokud ale počet x_3 není dělitelný osmi, zůstane nám nakonec jedna částečně zaplněná paleta. V ní bude volný prostor, do kterého se snažíme přednostně umístit co nejvíce krabic velikosti 40 cm a pak zbytek místa vyplnit krabicemi velikosti 20 cm. V tuto chvíli nemůžeme udělat nic lepšího, než si předem určit v závislosti na hodnotě zbytku p_3 po dělení osmi, jak se nám navýší volná kapacita x_1 a x_2 díky této poslední ne zcela zaplněné paletě. Určení těchto hodnot vyžaduje trochu prostorové představivosti, konkrétní číselné hodnoty najdete v přiloženém programu v seznamu x.

Zbytek řešení je pak již snadný. Porovnáme počet p_2 přepravovaných krabic velikosti 40 cm s volnou kapacitou x_2 a je-li vyšší, umístíme co největší počet krabic 40 cm do existujících palet. Pokud je hodnota p_2 nižší než x_2, umístíme do stávajících palet všechny krabice 40 cm a zbývající nevyužitý prostor v paletách $x_2 - p_2$ dáme k dispozici pro krabice 20 cm. To znamená, že volnou kapacitu x_1 můžeme zvýšit o $(x_2 - p_2) \times 8$. Následně porovnáme počet p_1 přepravovaných krabic velikosti 20 cm s volnou kapacitou x_1 a je-li vyšší, opět umístíme co největší počet krabic 20 cm do existujících palet. Je-li hodnota p_1 nižší než x_1, umístíme do stávajících

palet všechny krabice 20 cm. Nakonec se podíváme, zda nám zbyly ještě nějaké neumístěné krabice velikosti 40 cm nebo 20 cm. Pokud ano, spočítáme jejich celkový objem a z něj už snadno odvodíme počet palet, které pro tyto krabice musíme ještě přidat.

Popsaný algoritmus má konstantní časovou i prostorovou složitost. Nepotřebujeme v něm provádět žádné cykly, jejichž počet opakování by závisel na počtu přepravovaných krabic, nepotřebujeme ani žádnou datovou strukturu této velikosti. Ukázkový program v jazyce Python je jen přímou implementací popsaného postupu řešení. Podobně jako u úlohy s dělením pizzy je velmi jednoduchý a krátký.

```
p1, p2, p3, p4, p5, p6 = [int(_) for _ in input().split()]
# v pořadí krabice velikosti
# 20 cm, 40 cm, 60 cm, 80 cm, 100 cm, 120 cm

p = p6 + p5 + p4 + p3 // 8   # potřebný počet přepravních palet
if p3 % 8 > 0:
    p += 1

x = ((0,0),(37,19),(42,15),(47,11),(36,9),(41,5),(30,3),(19,1))
# navýšení volné kapacity (x1,x2) v závislosti na hodnotě p3 % 8

x1 = p5 * 91 + x[p3 % 8][0]
# volná kapacita pro krabice velikosti 20 cm
x2 = p4 * 19 + x[p3 % 8][1]
# volná kapacita pro krabice velikosti 40 cm

if p2 >= x2:
    p2 -= x2
else:
    p2, x1 = 0, x1 + (x2 - p2) * 8
if p1 >= x1:
    p1 -= x1
else:
    p1 = 0

z = p2 * 8 + p1
p += z // 216   # do palety se vejde 216 krabic velikosti 20 cm
if z % 216 > 0:
    p += 1

print(p)
```