Rozhledy matematicko-fyzikální

Stanislav Trávníček

Rozdělení napůl

Rozhledy matematicko-fyzikální, Vol. 87 (2012), No. 3, 18–22

Persistent URL: http://dml.cz/dmlcz/146480

Rozdělení napůl

Stanislav Trávníček, PřF UP, Olomouc

Abstract. The article deals with this problem: Given N objects of different masses, the task is to create a computer program which divides the set of all the objects into two groups so that the sums of the masses in the two groups are as close to each other as possible. The program is also supposed to determine the mass of each of the groups and to list the objects of each of the groups.

Pan Koumal se vydal na velký nákup. Byl úspěšný, a tak se mu v nákupním vozíku navršila hromada potřebných věcí, které si po zaplacení rozdělil do dvou tašek a vydal se na cestu domů. Za chvíli zjistil, že jedna taška je o dost těžší než ta druhá, a tak se zastavil a začal věci přerovnávat. Teď už to bylo lepší, ale stejně se po nějaké chvíli znovu zastavil a znovu obsah tašek přerovnával. To ho přimělo k zamyšlení, jak si měl nákup rozdělit, aby obě tašky byly skoro stejně těžké. Tento problém mu nedal pokoje a jelikož uměl trochu programovat pro počítač, řekl si, že se pokusí vyřešit jej ve spolupráci s počítačem. Napřed si však ujasnil, čeho chce dosáhnout.

Problém 1. *Máme N věcí o hmotnostech $M_1, M_2, \ldots, M_N$. Chceme je rozdělit na dvě skupiny, jejichž hmotnost by se lišila co nejméně, a chceme návod, kterou věc dát do které skupiny. Pokud je takových možností více, stačí jedna z nich. Také chceme znát hmotnost obou skupin a údaj, o kolik je jedna skupina těžší než druhá.*

Začátek programu (nazval jej *Vyvážení*) byl panu Koumalovi celkem jasný – musí se zadat vstupní data, tedy N (kolik je celkem věcí), a pak je třeba zadat a uložit hmotnosti jednotlivých věcí do pole (nazval je) M, tedy $M[I]$, $I = 1, 2, \ldots, N$. Také si uvědomil, že hmotnosti věcí nebude možno vyjádřit přirozeným číslem, takže počítal s tím, že pro prvky pole M zvolí typ *Real*.

Ke způsobu práce připravovaného programu si pan Koumal řekl, že nezpochybnitelný (tedy správný) výsledek dostane jen tehdy, když vyzkouší všechny možnosti rozdělení věcí na dvě skupiny, a vypočítal, že

N-tic jedniček a dvojek je 2^N. Dále řešil, jakým způsobem jednotlivé případy zaznamenávat, aby bylo jasné, která věc se má zařadit do 1. skupiny a která do 2. skupiny. Věděl, že musí k poli M zvolit další pole, nazval je *Ano*, a považoval za velmi zdařilé pravidlo, že by na místech příslušných k prvkům pole M byly jedničky či dvojky, které by značily číslo skupiny, kam příslušná věc patří. Musí tedy vytvořit z jedniček a dvojek všechny možné N-tice. Ale jak získat všechny případy?

Princip mu byl jasný: Seřadíme N-tice tak, že nejprve vezmeme ty, kde $Ano[1] = 1$ (to bude první polovina případů), a pak ty, kde $Ano[1] = 2$. V té první polovině vezmeme nejprve případy, kde $Ano[2] = 1$, a pak ty, kde $Ano[2] = 2$. Stejně budeme postupovat ve druhé polovině případů. Stejný postup zvolíme pro $Ano[3]$, $Ano[4]$ atd. až pro $Ano[N]$. Ale jak tento princip přenést do programu? Užití iterací typu

```
for I := 1 to 2 do
```

pan Koumal zavrhl, protože by jich muselo být N; to se však volí až za chodu programu. A tu ho napadla rekurze (pan Koumal nebyl v programování příliš zběhlý, takže ho ta rekurze napadla až po chvíli). Hledanou rekurzivní proceduru nazval `Rozloz` a hned si zapsal její řídicí část, která postupně zajistí vytvoření všech možných N-tic jedniček a dvojek:

```
procedure Rozloz(J: Integer)
begin
    Ano[J] := 1;
    Rozloz(J - 1);
    Ano[J] := 2;
    Rozloz(J - 1)
end;
```

Teď si pan Koumal uvědomil, že jednotlivé případy budou seřazeny jinak, než si řekl, tedy odzadu; nejprve budou případy, kdy $Ano[N] = 1$, atd., ale to na věci nic nemění. Pak na chvíli opustil úvod procedury a zamýšlel se nad tím, jaké akce a výpočty je třeba provádět v každém jednotlivém případu.

Nejprve zjistíme hmotnosti obou skupin; označil je *TSk1* a *TSk2* (aby mu označení napovědělo, že takové hmotnosti jsou právě teď) a pak absolutní hodnotu *TR* jejich rozdílu. Zjištění výsledné nejmenší hodnoty je klíčová akce, uložíme ji do proměnné *Rozdil*. Na počátku, uvažoval pan Koumal, dáme do proměnné *Rozdil* nesmyslně velikou hodnotu a pak se vždy budeme ptát, zda $TR < Rozdil$. Když ano, tak jsme narazili na

zatím nejlepší řešení, přesuneme obsah *TR* do proměnné *Rozdíl* a také musíme uchovat obsah pole *Ano* v dalším poli, nazvěme ho *TakJeTo*, a v *Sk1* a *Sk2* uchováme hmotnosti *TSk1*, *TSk2* obou skupin. Po průchodu všemi N-ticemi tak získáme všechny údaje požadované úlohou.

Takže posledním problémem pro pana Koumala byl krok od vytvoření N-tice uložené v poli *Ano* k jejímu zpracování. Ale to se nakonec ukázalo docela jednoduché. Jestliže se totiž v rekurzivním procesu narazí na požadavek *Rozloz*(0), znamená to, že se právě dospělo k *Ano*[1], tedy nová N-tice je v poli *Ano* již vytvořena, takže v této chvíli se přejde na zpracování nového případu. K tomu stačí nastavit podmínky pro J; pro $J > 0$ se nový případ tvoří, pro $J = 0$ se zpracuje a pak se použije rozhodování:

```
procedure Rozloz(J: Integer)
if J>0 then
begin
{ vytváření nového případu }
end
else begin
{ zpracování vytvořeného případu }
end
```

Samotný program už nedal žádnou práci: načtou se vstupy, zvolí se hodnota proměnné *Rozdil*, zavolá se rekurzivní procedura a vytisknou se výsledky.

Pan Koumal si však řekl, že by jeho program měl být co nejracionálnější. Tak si třeba uvědomil, že vlastně nepotřebuje všechny N-tice, protože každý případ se tak probírá dvakrát, jen se obě skupiny zamění. Stačilo by tedy vyšetřovat jen 2^{N-1} N-tic, tedy jen (tu správnou) polovinu případů. A tak si řekl: N-tou věc dám vždycky do 1. skupiny, tedy pevně zvolím $Ano[N] = 1$, takže budu vytvářet jen $(N-1)$-tice pro věci 1 až $N-1$. Avšak ještě nebyl spokojený. Napadlo ho totiž, že by se chod programu mohl ukončit okamžitě, pokud by v jeho průběhu nastala rovnost $Rozdil = 0$. Protože si dopředu stanovil, že stačí najít jen jedno optimální uspořádání, a pro $Rozdil = 0$ takové již má, bylo by proto vytváření dalších případů již zbytečné, takže procedura *Rozloz* se může zastavit, a to zcela jednoduše, když jako první dá příkaz:

```
if Rozdil = 0 then Exit;
```

Uveďme nyní výpis programu *Vyvážení*, jak jej vytvořil pan Koumal.

```
program Vyvazeni;
var
 N, I:  Integer;
 Rozdil, Sk1, Sk2: Real;
 Ano, TakJeTo: array [1..100] of Integer;
 M: array [1..100] of Real;

procedure Rozloz(J: Integer);
var
  TR, TSk1, TSk2: Real;
  I: Integer;
begin
  if Rozdil = 0 then Exit;
  if J > 0 then
  begin
   Ano[J] := 1;
   Rozloz(J-1);
   Ano[J] := 2;
   Rozloz(J-1);
  end
  else begin
   TSk1 := 0; TSk2 := 0; Ano[N] := 1;
   for I := 1 to N do
     if Ano[I] = 1 then TSk1 := TSk1 + M[I]
     else TSk2 := TSk2 + M[I];
   TR := Abs(TSk1 - TSk2);
   if TR < Rozdil then
   begin
     Rozdil := TR;
     Sk1 := TSk1; Sk2 := TSk2;
     TakJeTo := Ano
   end
  end
end;

begin {program}
 Write('Pocet veci: ');
 ReadLn(N);
```

```
 WriteLn('Hmotnosti veci:');
 for I:=1 to N do
   ReadLn(M[I]);
 Rozdil := 1000;
 Rozloz(N - 1);
 WriteLn('Hmotnost 1. skupiny je ',Sk1:0:3);
 for I := 1 to N do
   if TakJeTo[I] = 1 then Write(' ',I);
 WriteLn;
 WriteLn('hmotnost 2. skupiny je ',Sk2:0:3);
 for I := 1 to N do
   if TakJeTo[I] = 2 then Write(' ',I);
 WriteLn;
 Write('Rozdil je ');
 WriteLn(Rozdil:0:3);
 ReadLn
end
```

Pan Koumal měl velikou radost, že se mu stanovený problém podařilo zdárně vyřešit, i když mu bylo jasné, že v praxi, po nákupu v obchodě a při rozdělování věcí do dvou tašek, by takový postup nemohl uplatnit.

Viděli jsme, jak si počínal pan Koumal. Avšak vy, kteří umíte programovat lépe než on, se můžete pokusit najít alternativu jeho řešení nebo snad i zlepšit použitou rekurzivní proceduru.

(Autorkou ilustračního obrázku je Mgr. Jaroslava Čermáková.)