ПОДЕЛИ ПА ВЛАДАЈ

Зоран Будимац и Мирјана Ивановић, Нови Сад

Техника "подели па владај" (енгл. *divide and conquer*) је техника програмирања, односно конструкције алгоритама. Основна идеја ове технике је да се решење неког проблема P сведе на решење мањих проблема исте врсте. Потом се решења мањих проблема укомпонују у решење почетног проблема P.

У општем случају техника "подели па владај" се може описати на следећи начин у псеудо-Паскалу. Подразумевамо да је P проблем који треба да се реши, n његова величина, а R решење проблема. "Величину" проблема засад схватамо интуитивно – видећемо касније да се она често своди на димензије низа или матрице.

```
PROCEDURE Resi(P, n, VAR R);
IF n "довољно мало" THEN
   Израчунај R
ELSE BEGIN
   Подели проблем P на потпроблеме P1 ... Pk величина (редом) n1 ... nk, тако
   да је бар једно n1 ... nk мање од n
   Resi(P1, n1, R1); ... Resi(Pk, nk, Rk);
   Укомпонуј решења R1 ... Rk у решење R
END
```

Да би се проблем могао решавати техником "подели па владај", потребно је да испуњава неколико услова: 1) проблем мора имати лако мерљиву "величину"; 2) мора бити дељив на мање потпроблеме, тако да се сваки потпроблем од почетног разликује само по величини; 3) мора бити могуће укомпоновати решења потпроблема у решење почетног проблема; и 4) ако је проблем мали (тј. мале је величине) мора бити лако решив.

Техника "подели па владај", ако се добро примени, обично даје ефикасне алгоритме за решење конкретних проблема. У општем случају важи да је резултујући алгоритам ефикаснији, ако: а) су потпроблеми подједнаке величине и б) сваки потпроблем има што мање заједничких елемената са другим потпроблемима. Највећа ефикасност се дакле може очекивати ако се почетни проблем дели на једнаке и дисјунктне потпроблеме. Ово ипак није генерално правило – и ако су ова два услова испуњена резултујући алгоритам не мора бити нарочито ефикасан. За сваки конкретни алгоритам заснован на овој техници, морамо посебно анализирати ефикасност.

Техника "подели па владај", као и препознавање проблема који се њоме могу решавати се најбоље могу научити кроз праксу и велики број примера. Због тога се до краја овог текста концентришемо на примере употребе ове технике. Ефикасност појединих алгоритама ћемо изражавати "велико О" нотацијом. Ако кажемо да је рачунска сложеност неког алгоритма $O(f(n))$, то значи да је за довољно

велико n ($n > n_0$) број корака алгоритма пропорционалан са $f(n)$. Истичемо улогу константе n_0 – понекад се може десити да се алгоритам веће рачунске сложености за мале вредности n ($n \leq n_0$) извршава брже од алгоритма мање рачунске сложености.

Напоменимо на крају уводног дела да су процедуре којима се реализује техника "подели па владај" најчешће *рекурзивне*. За процедуру (или функцију) кажемо да је рекурзивна ако се у њеној дефиницији ("телу") користе позиви исте те процедуре (функције) – непосредно или посредно. Да би рекурзивно решење неког проблема било коректно, потребно је да обавезно садржи решење за једноставни случај проблема, те да случајеви који се решавају рекурзивним позивом "теже" ка једноставном случају (ови захтеви су аутоматски испуњени код технике "подели па владај"). "Класични" пример рекурзивне функције је функција за израчунавање факторијела за број n (у ознаци $n!$):

$$n! = \begin{cases} 1, & n = 1 \\ n \cdot (n-1)!, & n > 1 \end{cases}$$

Максимални елемент низа

Нека је дат низ `N` позитивних целих бројева и нека је потребно пронаћи његов максимални елемент. Овај проблем се може решити техником "подели па владај" јер задовољава сва четири услова (проверите). Низ `N` ћемо делити на две половине, пронаћи максимум у обе половине, а већи од њих ће бити максимум полазног низа. Следи решење у Паскалу, при чему је `Niz` декларисан као низ целих бројева у коме се налазе позитивни цели бројеви, `d` и `g` су доњи и горњи индекс низа између којих се тражи максимални елемент, а `maks` је пронађени максимум.

```
PROCEDURE Max(VAR N: Niz; d, g: INTEGER; VAR maks: INTEGER);
VAR pola: INTEGER;          (* polovina niza *)
    maksL, maksD: INTEGER; (* maksimumi leve i desne polovine niza *)
BEGIN
   IF d > g THEN            (* niz je ''prazan'' *)
      maks := 0
   ELSE IF d = g THEN       (* niz ima jedan element *)
      maks := N[d]
   ELSE BEGIN
      pola := (d+g) DIV 2;
      Max(N, d, pola, maksL);     (* maksimum leve polovine niza *)
      Max(N, pola+1, g, maksD);  (* maksimum desne polovine niza *)
      IF maksL > maksD THEN maks := maksL ELSE maks := maksD;
   END
END;
```

У решењу смо навели два једноставна случаја: кад је низ празан и кад има један елемент. Ово је неопходно јер почетни низ може бити непарне дужине, па током половљења може доћи до оба случаја. Уместо празног низа смо могли навести случај низа од два елемента, али је проналажење максималног елемента празног низа нешто једноставније (узмемо да је то, по дефиницији, 0).

Сортирање мешањем

Нека је дат низ N целих бројева и нека је потребно сортирати га. Техником "подели па владај" се сортирање може урадити на следећи начин: поделити низ на две половине, сортирати сваку половину посебно, те потом "помешати" обе сортиране половине на одговарајући начин како би се добио сортирани цео низ. Наводимо решење у Паскалу за сортирање низа N од индекса d до индекса g.

```
PROCEDURE MSort(VAR N: Niz; d, g: INTEGER);
VAR pola: INTEGER;              (* polovina niza *)
BEGIN
   IF d < g THEN BEGIN          (* niz ima vise od jednog elementa *)
      pola := (d+g) DIV 2;
      MSort(N, d, pola);        (* sortiranje leve polovine niza *)
      MSort(N, pola+1, g);      (* sortiranje desne polovine niza *)
      Pomesaj(N, d, pola, g)    (* dve polovine u sortirani niz *)
   END
END;
```

Ако је низ празан или има један елемент (тј. кад је d ≥ g) није потребно ништа радити – низ је тада већ сортиран. Процедура Pomesaj ће помешати леву (сортирану) половину низа N (од d до pola) са десном (сортираном) половином низа (од pola+1 до g) у коначно решење.

```
PROCEDURE Pomesaj(VAR N: Niz; d, pola, g: INTEGER);
VAR i, pom: INTEGER;
BEGIN
   IF (d <= pola) AND ((pola+1)<=g) THEN BEGIN (* polovine su ''pune'' *)
      IF N[d] <= N[pola+1] THEN      (* poredimo prve elemente *)
         Pomesaj(N, d+1, pola, g)    (* N[d] ostaje na svom mestu *)
      ELSE BEGIN
         pom := N[pola+1];
         FOR i := pola+1 DOWTO d+1 DO  (* pomeranje leve polovine *)
             N[i] := N[i-1];
         N[d] := pom;                  (* novi prvi element *)
         Pomesaj(N, d+1, pola+1, g)    (* mesamo ostatak *)
      END
   END
END;
```

Рачунска сложеност овог алгоритма је $O(n \log_2 n)$, што је неома ефикасно. Ако се дужина низа кога треба сортирати повећа 10 пута, време потребно за сортирање се повећа "само" 33 пута.

Брзо сортирање

Идеју "подели па владај" је могуће и на другачији начин применити на решење проблема сортирања низа. У досадашњим примерима је дељење проблема

на потпроблеме било врло једноставно, док се главна активност одвијала током компоновања решења. Могуће је ствари поставити и другачије - да се главна активност одвија током поделе на потпроблеме, док би тада компоновање решења било врло једноставно. Примењено на проблем сортирања, овај други приступ даје следећи поступак:

1. Одабери произвољни елемент `x` низа `N`;
2. Све елементе мање од `x` пребаци лево од `x` у низу `N`, а све елементе веће од `x` пребаци десно од `x` у низу `N`;
3. Кораке 1. и 2. понови за леви подниз (елемената мањих од `x`) и десни подниз (елемената већих од `x`);
4. Ако је низ празан или има један елемент, сортирање није потребно.

Овај алгоритам је познат под именом брзо сортирање (енгл. *quick sort*). Његова ефикасност зависи од избора елемента `x` – у најгорем случају (када се за `x` увек бира минимални или максимални елемент низа) рачунска сложеност овог алгоритма је $O(n^2)$. Показано је међутим, да је у просечном случају сложеност овог алгоритма $O(n \log_2 n)$.

Множење

По класичном алгоритму множења два броја x и y, број x се множи са сваком цифром броја y, добијају се парцијалне суме померене за одговарајући број места улево, које се на крају саберу дајући резултат множења. Рачунска сложеност овог алгоритма је $O(n^2)$, при чему је n број цифара већег броја.

Техника "подели па владај" се може успешно применити и на множење два броја, што резултује новим, мање познатим и ефикаснијим алгоритмом за множење. Основна идеја се састоји да се оба броја поделе на половине једнаке дужине (због једноставности подразумевамо да су бројеви исте дужине, n, и да је n степен двојке): $x = x_1 \cdot 10^{n/2} + x_0$ и $y = y_1 \cdot 10^{n/2} + y_0$. Тада је производ бројева x и y једнак броју $z_2 \cdot 10^n + z_1 \cdot 10^{n/2} + z_0$, при чему је:

$$\begin{aligned} z_2 &= x_1 \cdot y_1 \\ z_1 &= (x_1 \cdot y_0) + (x_0 \cdot y_1) \\ z_0 &= x_0 \cdot y_0 \end{aligned}$$

Када је проблем довољно мали (на пример, множење два *једноцифрена* броја) бројеви се директо измноже.

Алгоритам илуструјемо на примеру множења бројева $x = 1234$ и $y = 5678$. Даље је:

$$x = x_1 \cdot 10^2 + x_0 = \underbrace{12}_{x_1} \cdot 10^2 + \underbrace{34}_{x_0}$$
$$y = y_1 \cdot 10^2 + y_0 = \underbrace{56}_{y_1} \cdot 10^2 + \underbrace{78}_{y_0}$$

а одатле следи:

$$z_2 = x_1 \cdot y_1 = 12 \cdot 56 = 672$$
$$z_1 = (x_1 \cdot y_0) + (x_0 \cdot y_1) = (12 \cdot 78) + (34) = 2840$$
$$z_0 = (x_0 \cdot y_0) = 34 \cdot 78 = 2652$$

при чему би се двоцифрени бројеви при израчунавању z_0, z_1 и z_2 множили на аналоган начин, али те кораке због једноставности изостављамо. Резултат је тада

$$z = x \cdot y = \underbrace{672}_{z_2} \cdot 10^4 + \underbrace{2840}_{z_1} \cdot 10^2 + \underbrace{2652}_{z_0} = 7006652$$

Овим алгоритмом смо множење два n-тоцифрена броја свели на четири множења $(n/2)$-цифрена броја. Примећујемо такође да се проблем множења броја са 10^k своди на померање броја k места у лево, те да је сабирање два n-тоцифрена броја сложености $O(n)$. Имајући све то у виду, може се показати да је рачунска сложеност "подели па владај" множења $O(n^2)$, што је једнако класичном алгоритму.

Уколико међутим израчунавање z_1 у горњем алгоритму заменимо еквивалентном алтернативом:

$$z_1 = (x_1 + x_0) \cdot (y_1 + y_0) - z_2 - z_1$$

могуће је смањити број множења. Овом изменом смо множење два n-тоцифрена броја свели на *три* множења $(n/2)$-цифрена броја, али уз једно сабирање и два одузимања више него у претходном решењу. Како је сложеност сабирања и одузимања линеарна $(O(n))$, рачунска сложеност множења се смањује и сада је сложености $O(n^{1.59})$, што је значајно ефикасније од класичног алгоритма. Морамо напоменути да се за мале вредности n ова разлика неће видети, те да ће се у таквим случајевима класични алгоритам брже извршевати. За велике целе бројеве, наше побољшано "подели па владај" множење ће се извршавати значајно брже.

Задаци

1. Да ли је проналажење максималног елемента у низу техником "подели па владај" ефикасније од уобичајеног алгоритма, по коме се максималним елементом прогласи нека вредност, а потом се та вредност пореди редом са сваким елементом низа?

2. Уколико алгоритам сортирања мешањем модификујемо тако да се лева половина низа састоји само од једног (првог) елемента, а десна половина од свих осталих елемената, тада добијамо сортирање уметањем. Алгоритам се састоји од сортирања десног дела низа, па уметања преосталог елемента на одговарајуће место у сортирани низ. Реализовати сортирање уметањем. Каква је ефикасност сортирања уметањем у односу на сортирање мешањем?

3. Погледати задатак из програмирања П25 (овај број тангенте).

4. Уопштити "подели па владај" алгоритам множења, тако да бројеви не морају бити једнаке дужине и да дужина не мора бити степен двојке. Бројеве вишеструке прецизности представити низом цифара. Реализовати класични алгоритам множења оваквих бројева и побољшани "подели па владај" алгоритам. Експериментално испитати после које дужине бројева је "подели па владај" алгоритам бржи.

5. Написати програм користећи технику "подели па владај", који у опсегу бројева од a до b проналази најмањи број са задатом особином. Особина је представљена логичком функцијом F.

6. Да ли је израчунавање детерминанте матрице развојем по једној врсти (или колони) матрице и за сваки елемент врсте израчунавањем детерминанте минора, употреба технике "подели па владај"? Да ли је овакво израчунавање ефикасно? Зашто? Написати програм за израчунавање детерминанте матрице на другачији начин.

Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 1997/98 година