ОӘЖ 004.42

МІНСІЗ САНДАРДЫ JAVA ТІЛІНДЕ КӨПАҒЫНДЫ БАҒДАРЛАМАЛАУДЫ ҚОЛДАНЫП ӨНДІРУ

Жұмағұлов Марғұлан
zh.k.markus@gmail.com
Л.Н.Гумилев атындағы ЕҰУ «Математикалық және компьютерлік модельдеу» мамандығының 4-курс студенті
Ғылыми жетекшісі – А.С.Жумаханова, Л.Н.Гумилев атындағы ЕҰУ, «Математикалық және компьютерлік модельдеу» кафедрасының аға оқытушысы

Мінсіз сандар (кейбір дерек көздерінде керемет сандар, тамаша сандар деп те айтылады) дегеніміз – 1-ді қоса алғанда және сол санның өзінен басқа барлық оң бөлгіштерінің қосындысына тең натурал сан. Ең кіші мінсіз сан $6 = 1 + 2 + 3$. Екінші мінсіз сан $28 = 1 + 2 + 4 + 7 + 14$. Мінсіз сандардың құпиясы мен қасиеті тек математикада ғана емес, мәдениет пен дінде де кеңінен таралған. Мінсіз сандар тарихына қысқаша тоқталсақ.

Жұп мінсіз сандарды іздеу алгоритмі Евклидтің «Начала» кітабында сипатталған болатын. Кітапта Евклид егер $2^p - 1$ саны (Мерсенн сандары) жай сан болса, онда $2^{p-1}(2^p - 1)$ саны мінсіз болатындығы жазылған. Осы формула арқылы Евклид үшінші *(496)* және төртінші *(8128)* мінсіз санды тапты. Кейіннен, Леонард Эйлер барлық жұп мінсіз сандар Евклид сипаттаған түрде болатындығын дәлелдеді. Бүгінгі таңда Мерсенн сандары арқылы анықталған 51 мінсіз сан белгілі. Тақ мінсіз сандар әлі күнге дейін анықталмған. Алайда ондай сандардың жоқ екендігі де дәлелденбеген.

Мінсіз сандарды Java бағдарламалау тілі көмегімен анықтайық. Алдымен іздеу алгоритмін құрамыз. Негізгі іздеу құралы ретінде Евклидтің формуласын қолданамыз.

Қадам 1. p – саны ретінде 2-ден бастап *n*-санына дейінгі сандар тізбегі қабылданатын цикл құру.

Қадам 2. $2^{p-1}(2^p - 1)$ формуласына *Қадам 1*-де қабылданған *p* санын қойып, нәтижені жаңа *num* айнымалыға меншіктеу.

Қадам 3. Қадам 2-де алынған *num* айнымалысын 1-ден бастап *num*-ға дейінгі барлық бүтін сандарға бөліп, бүтін бөлінген жағдайда олардың қосындысын алып, оны жаңа *sum* айнымалысына меншіктейтін шартты беру.

Қадам 4. num және *sum* сандарын салыстырушы шартты беру. Егер тең болса, санды экранға шығарады. Кері жағдайда *Қадам 1*-ге барып цикл келесі *p* санын тексеруге көшеді.

ЭЕМ-дердің қуаттылықтары әртүрлі болғандықтан және уақытты үнемдеу үшін программаға ағындарды енгізейік. Сонымен қатар, әр ағын қанша уақытта есептеу жүргізетіндігін білу үшін уақытты қосайық. *p* санын 20-ға дейін деп алайық, ал ағында 5 бөлікке бөлейік. Программа коды келесідегідей жазылады.

```
import java.lang.Thread; import java.lang.Math;
public class Numbers {
public static void main(String[] args) {
System.out.println("\n\tСовершенные числа: ");
PerfectNumbers_1 output_1 = new PerfectNumbers_1();
output_1.start();
PerfectNumbers_2 output_2 = new PerfectNumbers_2();
output_2.start();
PerfectNumbers_3 output_3 = new PerfectNumbers_3();
output_3.start();
PerfectNumbers_4 output_4 = new PerfectNumbers_4();
output_4.start();
PerfectNumbers_5 output_5 = new PerfectNumbers_5();
output_5.start();}}
class PerfectNumbers_1 extends Thread {
public void run() {
long startTime = System.currentTimeMillis();
for (int p = 2; p <= 15; p++) {
int sum = 0; double num;
num = Math.pow(2, p - 1) * (Math.pow(2, p) - 1);
for (int j = 1; j < num; j++) {
if(num % j == 0)  {
sum += j;}}
if (sum == num) {
System.out.println("\t" + sum);}}
```

```
long stopTime = System.currentTimeMillis();
long elapsedTime = stopTime - startTime;
System.out.println("\tExecution time: " + elapsedTime + "ms or " + elapsedTime * 0.001 + "s or " +
elapsedTime * 0.001 / 60 + "m\n");}}
class PerfectNumbers_2 extends Thread {
public void run() {
long startTime = System.currentTimeMillis();
int p = 16; int sum = 0; double num;
num = Math.pow(2, p - 1) * (Math.pow(2, p) - 1);
for (int j = 1; j < num; j++) {
if(num % j == 0)  {
sum += j;}}
if (sum == num) {
System.out.println("\t" + sum);}
System.out.println("\tEnd of the 16th iteration");
long stopTime = System.currentTimeMillis();
long elapsedTime = stopTime - startTime;
System.out.println("\tExecution time: " + elapsedTime + "ms or " + elapsedTime * 0.001 + "s or " +
elapsedTime * 0.001 / 60 + "m\n");}}
class PerfectNumbers_3 extends Thread {
public void run() {
long startTime = System.currentTimeMillis();
int p = 17; long sum = 0; double num;
num = Math.pow(2, p - 1) * (Math.pow(2, p) - 1);
for (double j = 1; j < num; j++) {
if(num % j == 0)  {
sum += j;}}
if (sum == num) {
System.out.println("\t" + sum);}
System.out.println("\tEnd of the 17th iteration");
long stopTime = System.currentTimeMillis();
long elapsedTime = stopTime - startTime;
System.out.println("\tExecution time: " + elapsedTime + "ms or " + elapsedTime * 0.001 + "s or " +
elapsedTime * 0.001 / 60 + "m\n");}}
class PerfectNumbers_4 extends Thread {
public void run() {
long startTime = System.currentTimeMillis();
int p = 18; long sum = 0; double num;
num = Math.pow(2, p - 1) * (Math.pow(2, p) - 1);
for (double j = 1; j < num; j++) {
if(num % j == 0)  {
sum += j;}}
if (sum == num) {
System.out.println("\t" + sum);}
System.out.println("\tEnd of the 18th iteration");
long stopTime = System.currentTimeMillis();
long elapsedTime = stopTime - startTime;
System.out.println("\tExecution time: " + elapsedTime + "ms or " + elapsedTime * 0.001 + "s or " +
elapsedTime * 0.001 / 60 + "m\n");}}
class PerfectNumbers_5 extends Thread {
public void run() {
long startTime = System.currentTimeMillis();
```

```
int p = 19; long sum = 0; double num;
num = Math.pow(2, p - 1) * (Math.pow(2, p) - 1);
for (double j = 1; j < num; j++) {
if(num % j == 0)  {
sum += j;}}
if (sum == num) {
System.out.println("\t" + sum);}
System.out.println("\tEnd of the 19th iteration");
long stopTime = System.currentTimeMillis();
long elapsedTime = stopTime - startTime;
System.out.println("\tExecution time: " + elapsedTime + "ms or " + elapsedTime * 0.001 + "s or " +
elapsedTime * 0.001 / 60 + "m\n");}}
```

Нәтижесі.

```
Совершенные числа:
6
28
496
8128
33550336
Execution time: 7789ms or 7.789000000000001s or 0.12981666666666666m

End of the 16th iteration
Execution time: 27502ms or 27.502s or 0.45836666666666664m

8589869056
End of the 17th iteration
Execution time: 102663ms or 102.663s or 1.71105m

End of the 18th iteration
Execution time: 373330ms or 373.33s or 6.222166666666666m

137438691328
End of the 19th iteration
Execution time: 1346600ms or 1346.6000000000001s or 22.443333333333335m
```

Корытындылай келе, программа *p* = 19 дейінгі мінсіз сандарды жуық шамамен 23 минутта есептеп шығарды. Сонымен қатар, әр ағынның есептеуді қанша минутта жүргізгендігін де көре аламыз. Мінсіз сандарды Java бағдарламалау тілі арқылы анықтау есептеу барысын жеңілдетіп қана қоймай, әртүрлі аралықты беру арқылы нәтижеге қол жеткізуге мүмкіндік береді.

Список использованных источников

1. Шилдт, Герберт Java 8. Руководство для начинающих. - М.: Вильямс, 2015. - 720 с.
2. Брайон Гоете. Праллельное программирование в Java на практике. Перевод Сорокина А.В. Екатеринбург, июнь 2018.
3. https://www.examclouds.com/ru/exam/java-core-russian