

Programowanie III

Dokumentacja projektu “Kurierzy”

Michał Pawlak

Politechnika Śląska

Wydział Matematyki Stosowanej, Informatyka

Semestr III, grupa 1B

1. Temat projektu

Zadanie “Kurierzy”

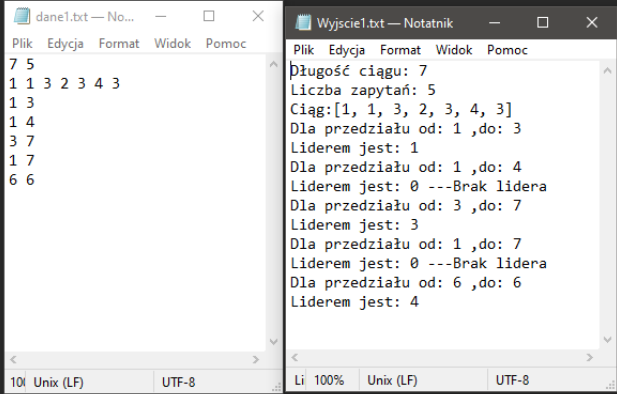
Bajtazar pracuje w firmie BAJ sprzedającej gry komputerowe. Firma BAJ współpracuje z wieloma firmami kurierskimi, które dostarczają sprzedawane gry klientom firmy BAJ. Baj-tazar prowadzi kontrolę tego, jak przebiegała współpraca firmy BAJ z firmami kurierskimi. Ma on listę kolejno wysłanych paczek, wraz z informacją o tym, która firma kurierska dostarczyła którą paczkę. Interesuje go, czy któraś z firm kurierskich nie uzyskała niezasłużonej przewagi nad innymi firmami kurierskimi. Jeżeli w jakimś przedziale czasu określona firma kurierska dostarczyła więcej niż połowę wysłanych wówczas paczek, to powiemy, że firma ta dominowała w tym czasie. Bajtazar chce stwierdzić, czy w określonych przedziałach czasu jakieś firmy kurierskie dominowały, a jeśli tak, to które to były firmy. Pomóż Bajtazarowi! Napisz program, który będzie znajdował dominującą firmę lub stwierdzi, że żadna firma nie dominowała.

2. Opis pobieranych danych przez program:

Program pobiera pliki .txt lub dane wpisywane przez użytkownika w konsoli. W celu poprawnego działania programu pliki powinny zawierać tylko liczby całkowite oddzielone znakami białymi.

3. Opis otrzymywanych rezultatów:

```
=====
Wybierz opcję: 1
=====
Podaj długość ciągu: 7
Podaj ilość zapytań: 5
t[0] = 1
Przeszukiwanie ciągu od pozycji: 1
Do pozycji: 1
Liderem jest: 1
=====
Kurierzy
=====
1. Konsola
2. Test plików
3. Wyjście
=====
Wybierz opcję: 3
=====
Dane wczytane z: C:\Users\blumi\Desktop\kur_prog\dane1.txt
Dane zapisane w: C:\Users\blumi\Desktop\kur_prog\Wyjscie1.txt
Dane wczytane z: C:\Users\blumi\Desktop\kur_prog\dane2.txt
Dane zapisane w: C:\Users\blumi\Desktop\kur_prog\Wyjscie2.txt
Dane wczytane z: C:\Users\blumi\Desktop\kur_prog\dane3.txt
Dane zapisane w: C:\Users\blumi\Desktop\kur_prog\Wyjscie3.txt
Dane wczytane z: C:\Users\blumi\Desktop\kur_prog\dane4.txt
Dane zapisane w: C:\Users\blumi\Desktop\kur_prog\Wyjscie4.txt
Dane wczytane z: C:\Users\blumi\Desktop\kur_prog\dane5.txt
Dane zapisane w: C:\Users\blumi\Desktop\kur_prog\Wyjscie5.txt
=====
Kurierzy
=====
1. Konsola
2. Test plików
3. Wyjście
=====
Wybierz opcję: 3
=====
Process finished with exit code 0
```



Wpisując dane z klawiatury użytkownik jest pytany o kolejne liczby, przy czym t[0] jest traktowane jako pozycja nr 1.

Testy wykonywane na plikach są wypisywane w plikach wyjściowych, ścieżki tych plików są wyświetlane w konsoli.

4. Zastosowany algorytm:

a) Słowny opis wykorzystanego algorytmu

Mamy dany ciąg składający się z n elementów. Musimy odpowiedzieć na m zapytań o lidera w pewnym fragmencie ciągu, przy czym liderem nazywamy element, który występuje w danym fragmencie więcej niż połowę razy.

Z pobranych danych wyciągamy pierwszą liczbę n - odpowiadającą za długość ciągu, oraz drugą liczbę m - odpowiadającą za ilość zapytań.

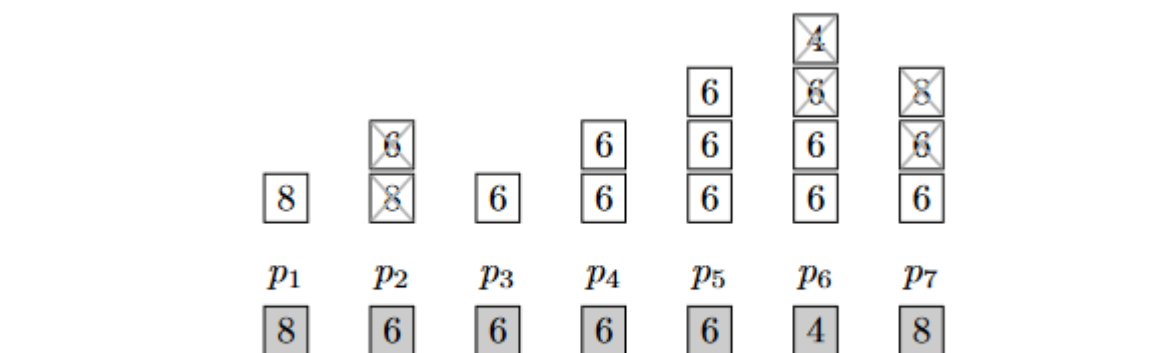
Kolejne n liczb są ciągiem w którym będziemy wyszukiwać lidera, następne m*2 liczb odpowiada za poszczególne zapytania.

Każde zapytanie określa początek i koniec przeszukiwanego fragmentu ciągu.

Wyszukując lidera korzystamy z nieskomplikowanego algorytmu.

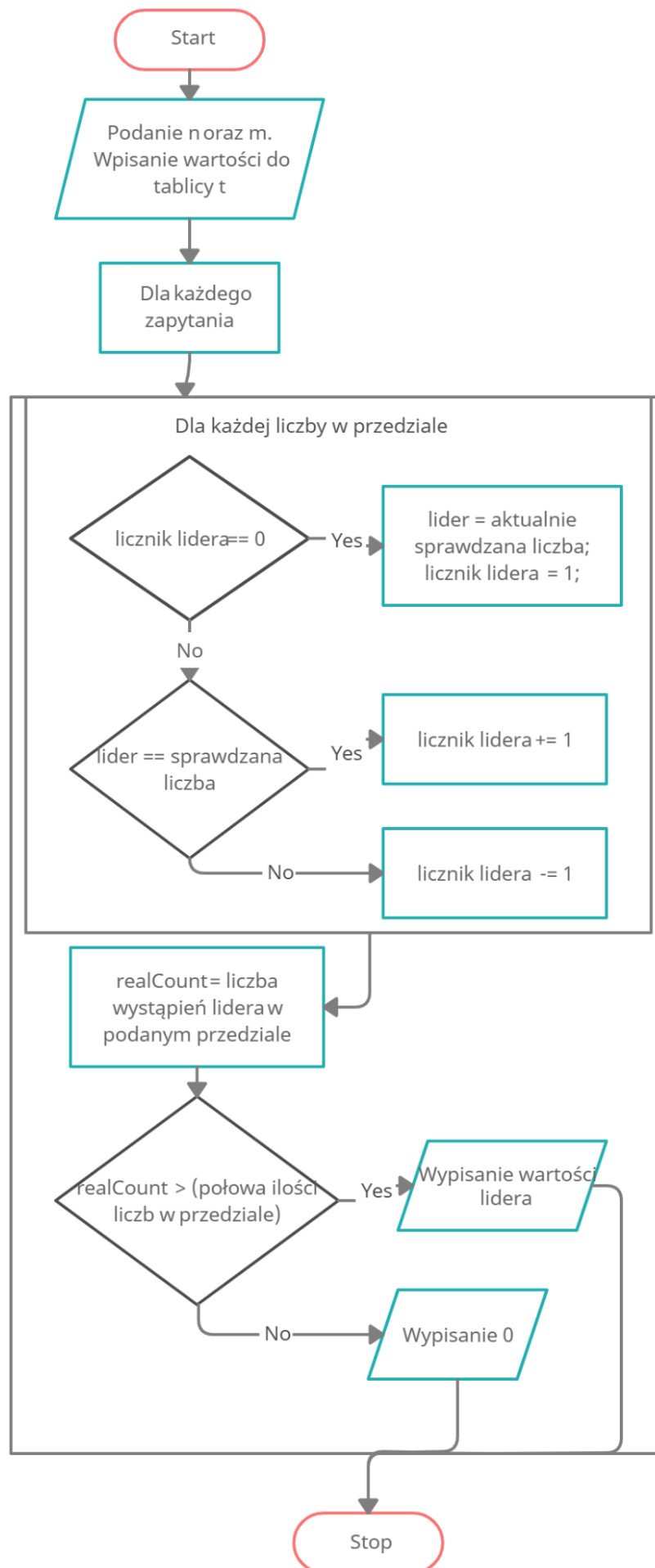
Jeśli nie mamy potencjalnego lidera, zostaje nim następna sprawdzona liczba i do licznika wystąpień lidera przypisywana jest wartość 1. W przeciwnym wypadku, jeśli następna liczba jest taka sama jak lider, to zwiększamy licznik wystąpień lidera. W innym wypadku licznik wystąpień spada o 1 (licznik nie osiąga wartości ujemnych). Po przeszukaniu w ten sposób podanego fragmentu sprawdzane jest czy licznik wszystkich wystąpień lidera jest większy od połowy ilości wszystkich liczb (w rozpatrywanym przedziale). Jeśli warunek jest spełniony, zwracana jest wartość lidera. W innym wypadku lider nie istnieje, zostaje zwracana wartość 0.

Obraz poglądowy opisanego algorytmu:



Rys. 1: Przykład działania algorytmu wyznaczania kandydata na lidera za pomocą stosu dla ciągu 8, 6, 6, 6, 6, 4, 8.

b) Schemat blokowy algorytmu



c)Zastosowanie metody (wpisywanie w konsoli)

```
static final int MAXN = 500005;
static void KurierzyMan(){
    Scanner sc = new Scanner(System.in);
    int[] t = new int[MAXN];
    int n, m;
    try{System.out.print("Podaj długość ciągu: ");
        n = sc.nextInt();
        System.out.print("Podaj ilość zapytań: ");
        m = sc.nextInt();
        for(int i = 0; i < n; i++){
            System.out.print("t["+ i +"] = ");
            t[i]=sc.nextInt(); }
        for(int i = 0; i < m; i++) {
            int a, b;
            System.out.print("Przeszukiwanie ciągu od pozycji: ");
            a = sc.nextInt() - 1;
            System.out.print("Do pozycji: ");
            b = sc.nextInt() - 1;
            int leader = -1, leaderCount = 0;
            for (int j = a; j <= b; j++) {
                if (leaderCount == 0) {
                    leader = t[j];
                    leaderCount = 1;
                } else {
                    if (leader == t[j])
                        leaderCount++;
                    else
                        leaderCount--;
                }
            }
            int realCount = 0;
            for (int j = a; j <= b; j++)
                if (t[j] == leader)
                    realCount++;
            if (realCount > (b - a + 1) / 2)
                System.out.println("Liderem jest: " + leader);
            else
                System.out.println("Liderem jest: " + 0 + " ---Brak lidera");
        }}catch (Exception e){e.printStackTrace();
        System.out.println("Błędne dane wejściowe!");}}
```

(wczytanie z pliku)

```
public static int num = 0;
static void KurierzyFile(File plik){
    num++;
    File f = new File(plik.getName());
    System.out.println("Dane wczytane z: " +plik.getAbsolutePath());
    try {
        File del = new File("Wyjscie"+num+".txt");
        del.delete();
        File Out = new File("Wyjscie"+num+".txt");
        if (Out.createNewFile()) {
            System.out.println("Dane zapisane w: " + Out.getAbsolutePath());
            FileWriter myWriter = new FileWriter((String)Out.getName());
            Scanner s = new Scanner(f);
            ArrayList<Integer> l = new ArrayList<Integer>();
            while (s.hasNext()) {l.add(s.nextInt());}
            int n = l.get(0);
            int m = l.get(1);
            //System.out.println("Długość ciągu: " + n);
            myWriter.write("Długość ciągu: " + n+"\n");
            //System.out.println("Liczba zapytań: " + m);
            myWriter.write("Liczba zapytań: " + m+"\n");
            ArrayList<Integer> ciag = new ArrayList<Integer>(l.subList(2, n + 2));
            //System.out.println("Ciąg:" + ciag);
            myWriter.write("Ciąg:" + ciag+"\n");
            for (int i = 0; i < m * 2; i += 2) {
                int a = l.get(2 + n + i) - 1;
                int b = l.get(3 + n + i) - 1;
                //System.out.print("Dla przedziału od: " + (a + 1) + " ,do: " + (b + 1) + "\n");
                myWriter.write("Dla przedziału od: " + (a + 1) + " ,do: " + (b + 1) + "\n");
                int leader = -1, leaderCount = 0;
                for (int j = a; j <= b; j++) {
                    if (leaderCount == 0) {
                        leader = ciag.get(j);
                        leaderCount = 1;
                    } else {
                        if (leader == ciag.get(j))
                            leaderCount++;
                        else
                            leaderCount--;
                    }
                }
                int realCount = 0;
                for (int j = a; j <= b; j++)
                    if (ciag.get(j) == leader) realCount++;
                if (realCount > (b - a + 1) / 2) {
                    //System.out.println("Liderem jest: " + leader);
                    myWriter.write("Liderem jest: " + leader+"\n");
                } else {
                    //System.out.println("Liderem jest: " + 0 + " ---Brak lidera");
                    myWriter.write("Liderem jest: " + 0 + " ---Brak lidera"+"\\n");}}
                myWriter.close();
            }else{System.out.println("Plik już istnieje");}
        }catch (IOException e) {
            e.printStackTrace();
            System.out.println("Error obsługi pliku");
        }catch (Exception e){
            e.printStackTrace();
            System.out.println("Błąd Programu");}}}
```

5. Testy na poprawność działania programu:

Możliwe błędy pobierania danych z plików lub z klawiatury są wyłapywane przez program. Natomiast zgodność wyników była sprawdzana za pomocą plików testowych załączonych na stronie olimpiady informatycznej (https://oi.edu.pl/I/21oi_ksiazeczka/).

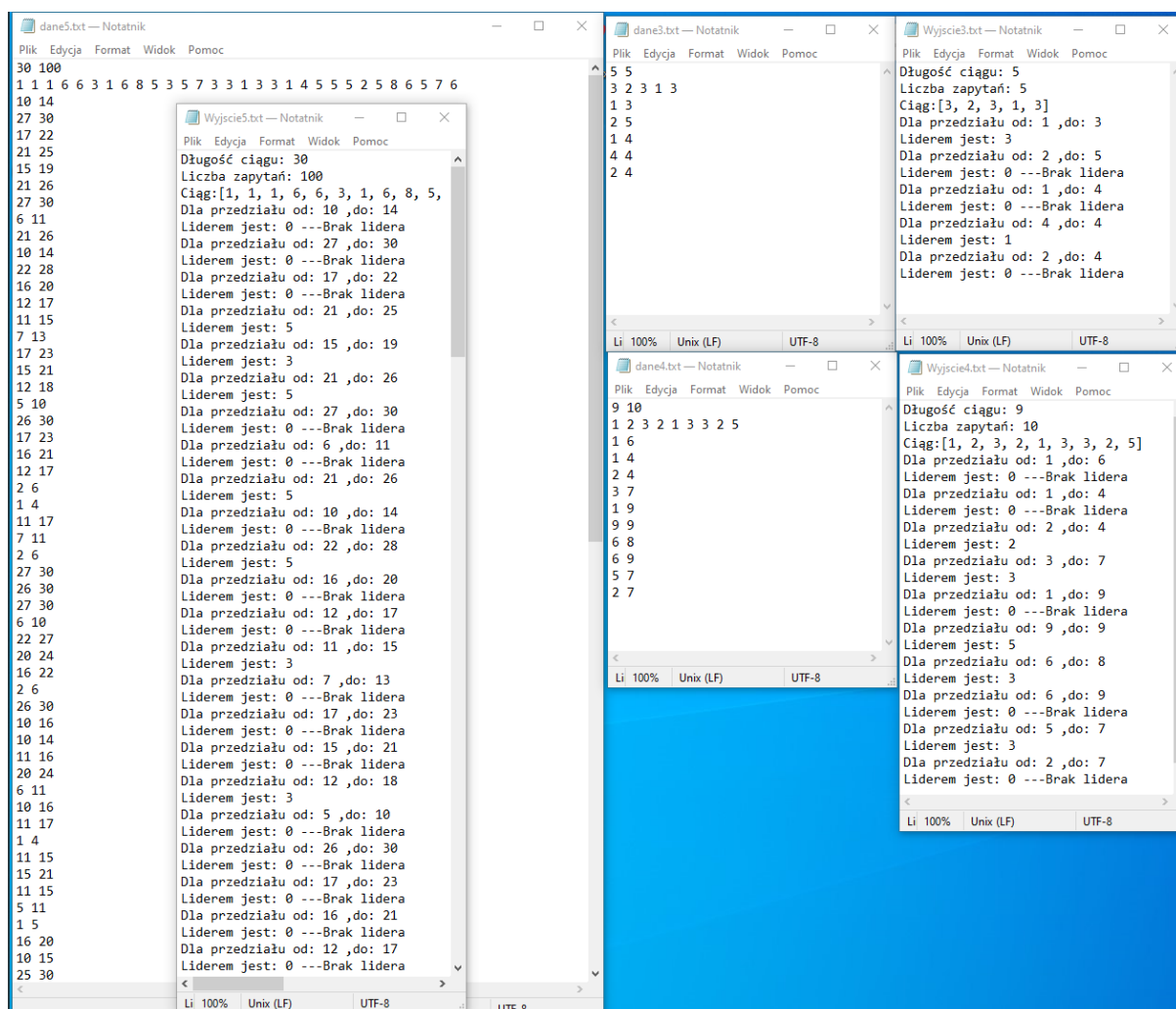
Przykład

Dla danych wejściowych:

```
7 5
1 1 3 2 3 4 3
1 3
1 4
3 7
1 7
6 6
```

poprawnym wynikiem jest:

```
1
0
3
0
4
```



6. Wnioski:

Samo zadanie nie było zbyt skomplikowane w implementacji. Problem zadania jest stosunkowo prosty do zrozumienia, dzięki czemu można napisać prototypowy kod i od niego rozbudowywać projekt.

Dzięki projektowi utrwaliłem umiejętności programowania w języku Java.

Niedociągnięciami tego projektu z pewnością byłby czas wykonania przy dużo dłuższych ciągach ($n > 100000$), gdzie rozwiązaniem tego problemu byłoby znalezienie lepszego algorytmu. Innym problemem jaki by się ujawnił byłby problem przejrzystości plików wyjściowych, gdzie ilość wierszy byłaby zbyt duża do efektywnego przejrzania.