Opracowanie: TAS
Taśma

1 Rozwiązanie I

Najprostszy sposób rozwiązania zadania wygląda tak:

- sprawdź wszystkie możliwe pary liczb;
- dla każdej, która zawiera **różne** liczby, zanotuj ich odległość;
- wybierz największą z zanotowanych odległości.

Aby uniknąć spamiętywania wszystkich „notowanych" odległości, możemy na bieżąco aktualizować największą zanotowaną odległość. Oto pseudokod programu realizującego to podejście:

```
wczytaj(m);
for test := 1 to m do
begin
    wczytaj(n);
    wczytaj(a[]);
    odleglosc := 0;
    for i := 1 to n do
    begin
        for j := (i + 1) to n do
            if a[i] <> a[j] then
                odleglosc := max(odleglosc, j - i);
    end
    if odleglosc = 0 then
        wypisz("BRAK");
    else
        wypisz(odleglosc);
end
```

Główny problem, jaki powstaje przy takim bezpośrednim podejściu, to złożoność czasowa, czyli szacowany czas działania algorytmu. Używamy w nim bowiem dwóch pętli, które przebiegają po wszystkich parach liczb, a więc wykonujemy $\frac{n\cdot(n-1)}{2}$, czyli co do rzędu wartości $O(n^2)$ kroków. Biorąc pod uwagę, że n może mieć wartość 100 000 — nasz algorytm wykona wtedy około $10 \cdot (10^5)^2 = 10^{11}$ kroków, podczas gdy należy oczekiwać, że w limicie czasowym mieszczą się rozwiązania wykonujące ok. 10^7 do 10^8 kroków. Tak więc to rozwiązanie jest zbyt wolne.

Punktacja przewidziana dla tego rozwiązania to 40%.

2 Rozwiązanie II (wzorcowe)

Aby napisać efektywniejszy program, wystarczy poczynić pewną obserwację. Otóż wystarczy rozpatrywać pary liczb:

- pierwsza liczba z dowolną inną;
- ostatnia liczba z dowolną inną.

Jeżeli bowiem rozwiązaniem byłaby inna para liczb, tzn. taka para (a_i, a_j), dla której $1 < i < j < n$, to wtedy wszystkie liczby $a_1, a_2, \ldots, a_{i-1}$ byłyby równe a_j, bo gdyby któraś z nich była różna od a_j, to w parze z nią byłaby lepszym rozwiązaniem niż (a_i, a_j). Rozumując symetrycznie, otrzymujemy, że liczby $a_{j+1}, a_{j+2}, \ldots, a_n$ są wszystkie równe a_i. W szczególności więc pokazaliśmy, że $a_1 = a_j$ oraz $a_i = a_n$. Zarazem jednak $a_i \neq a_j$, gdyż para (a_i, a_j) jest rozwiązaniem. W takim razie $a_1 \neq a_n$, więc tak naprawdę optymalnym rozwiązaniem jest para (a_1, a_n). Sprzeczność. A więc nasze założenie o tym, że rozwiązaniem może być para (a_i, a_j) spełniająca $1 < i < j < n$, jest błędne.

Dobrze. A zatem wiemy, że wystarczy sprawdzić tylko część par liczb. Ile ich jest? Policzmy:

- pierwsza liczba z dowolną inną — $n - 1$ par;
- ostatnia liczba z dowolną inną — $n - 1$ par.

Tak więc wystarczy rozpatrzyć $2n - 2$ pary (tak naprawdę jedna z nich pojawia się w obu powyższych grupach). Stąd złożoność czasowa tego rozwiązania wynosi $O(n)$. W praktyce oznacza to, że skoro $n \leqslant 100\,000$, to algorytm nie wykona więcej niż $10 \cdot 10^5 = 10^6$ kroków, co nas całkowicie satysfakcjonuje.

```
wczytaj(m);
for test := 1 to m do
begin
    wczytaj(n);
    wczytaj(a[]);
    odleglosc := 0;
```

```
    for i := 1 to n - 1 do
    begin
        if a[1] <> a[i + 1] then
            odleglosc := max(odleglosc, i);
        if a[i] <> a[n] then
            odleglosc := max(odleglosc, n - i);
    end
    if odleglosc = 0 then
        wypisz("BRAK");
    else
        wypisz(odleglosc);
end
```