1. Treść zadania

Dla tablicy o wymiarze n i elementach będących liczbami całkowitymi proszę napisać algorytm:

9.1 – wyznaczającą liczbę wystąpień minimalnego elementu w tablicy.

9.2 – wyznaczającą maksymalną odległość między elementami o minimalnej wartości (uwaga: minimalna wartość w tablicy jest jedna ale może wystąpić kilkakrotnie).

Np. dla tablicy {1, 4, 1 , 1, 7, 2, 3} minimalna wartość wynosi 1 i występuje 3 razy.

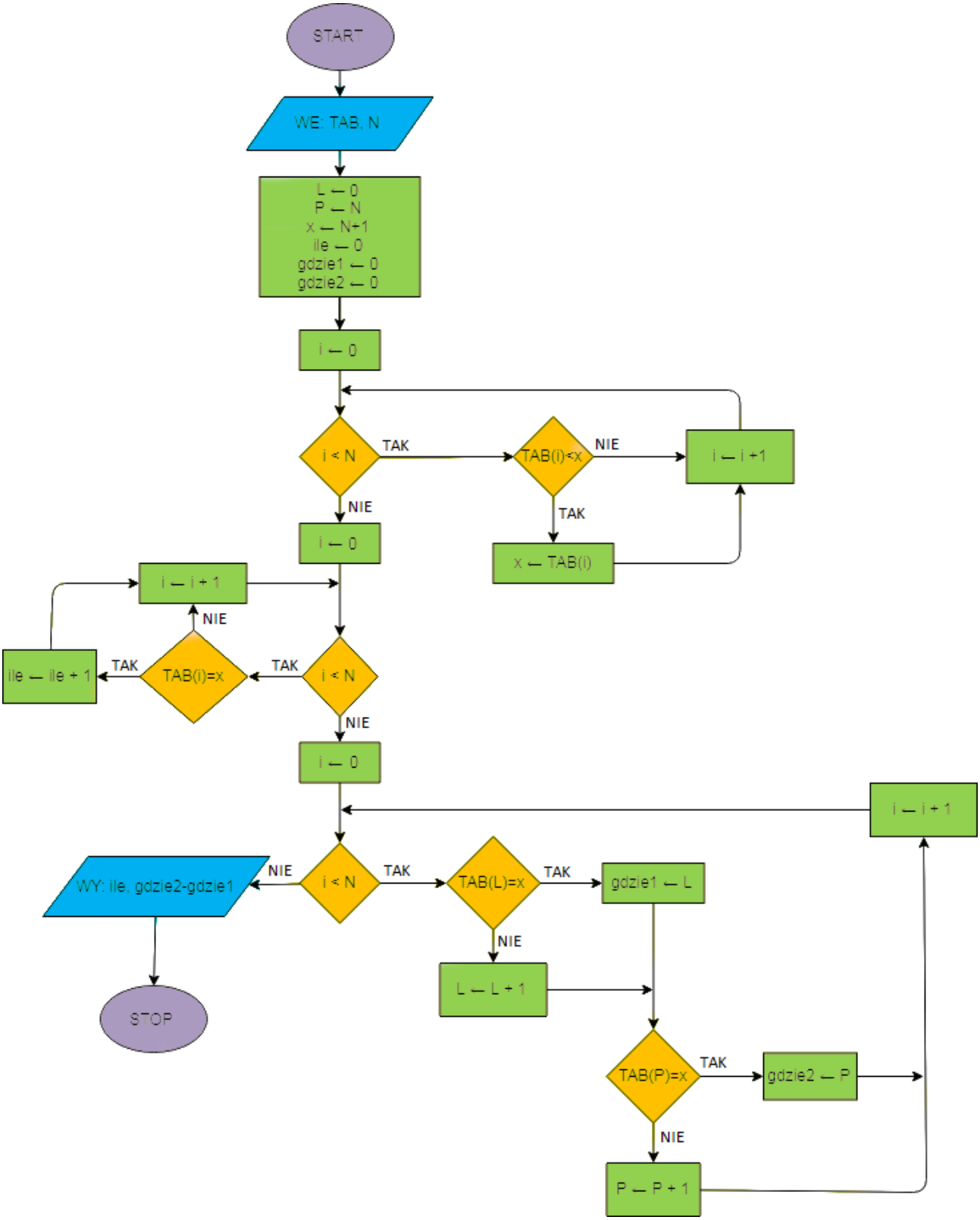
Maksymalna odległość między nimi wynosi 3 (różnica między indeksami ostatniego i pierwszego wystąpienia wartości 1). Jeśli wartość minimalna występuje jednokrotnie to maksymalna odległość wynosi 0.

2. Opis słowny algorytmu

Algorytm tworzy tablice ‘N’ elementowej długości następnie przypisuje wartości ‘x’ największą wartość po czym porównuje ją z każdym z elementów tablicy w poszukiwaniu wartości najmniejszej po jej znalezieniu przechodzi do następnego kroku którym jest zliczenie ile razy dana wartość w tabeli występuje. Ostatnim działaniem jest sprawdzenie najbliższego indeksu z najmniejsza wartością do strony lewej (L) i prawej (P) po czym odjęcie większego indeksu od mniejszego w celu wyliczenia najdalszej odległości między nimi.

Kod napisany w C++ jest dodatkowo urozmaicony o polecenia i opisy wyników w celu uporządkowania wprowadzanych i wyświetlanych danych oraz umożliwia wybór największej wartości do wylosowania.

3. Schemat blokowy



Schemat blokowy algorytmu

4. Symulacja działania algorytmu dla przykładowych danych WE

T	N	x	i	i<N	Tab(i)<x	x	i
2, 4, 6 , 8 ,2 ,9 ,3 ,3 ,4 ,5	10	11	0	TAK	2<11 TAK	2	1
		2	1	TAK	4<2 NIE	2	2
		2	2	TAK	6<2 NIE	2	3
			3	TAK	6<2 NIE	2	4
			4	TAK	2<2 NIE	2	5
			5	TAK	9<2 NIE	2	6
			6	TAK	3<2 NIE	2	7
			7	TAK	3<2 NIE	2	8
			8	TAK	4<2 NIE	2	9
			9	TAK	5<2 NIE	2	10
			10	NIE			

T	N	x	i	i<N	Tab(i)=x	ile	i
2, 4, 6 , 8 ,2 ,9 ,3 ,3 ,4 ,5	10	2	0	TAK	TAK	1	1
			1	TAK	NIE		2
			2	TAK	NIE		3
			3	TAK	NIE		4
			4	TAK	TAK	2	5
			5	TAK	NIE		6
			6	TAK	NIE		7
			7	TAK	NIE		8
			8	TAK	NIE		9
			9	TAK	NIE		10
			10	NIE			

T	N	x	i	i<N	L	Tab(L)=x	gdzie1	P	Tab(P)=x	gdzie2	i
2, 4, 6 , 8 ,2 ,9 ,3 ,3 ,4 ,5	10	2	0	TAK	0	2=2 TAK	0	9	5=2 NIE		1
			1	TAK				8	4=2 NIE		2
			2	TAK				7	3=2 NIE		3
			3	TAK				6	3=2 NIE		4
			4	TAK				5	9=2 NIE		5
			5	TAK				4	2=2 TAK	4	6
			6	TAK							7
			7	TAK							8
			8	TAK							9
			9	TAK							10
			10	NIE							

5. Zapis algorytmu w języku SCILAB

```
N = 10
TAB = int(N*rand(1,N))
disp(TAB)
L = 1
P = N
x = N+1
ile = 0
gdzie1 = 0
gdzie2 = 0

for i=1 : N
    if TAB(i)<x then
        x = TAB(i)
    end
end

for i=1 : N
    if TAB(i)==x then
        ile=ile+1
    end
end

for i=1 : N
    if TAB(L) == x then
        gdzie1=L
    else
        L=L+1
    end
    if TAB(P) == x then
        gdzie2=P
    else
        P=P-1
    end
end

disp(ile)
disp(gdzie2-gdzie1)
disp(timer())

```

6. Kod C++

```cpp
#include<iostream>
#include<ctime>
#include<random>

using namespace std;

int n;
int t[1000];

int main() {
    cout<<"Podaj ilosc liczb w tablicy: ";
    cin >> n;
    srand(time(NULL));
    int L = 0;
    int P = n-1;
    int a;
    cout<<"Podaj do jakiej liczby beda losowane wartosci: ";
    cin >> a;
    int x = a+1;
    int ile = 0;
    int gdzie1 = 0;
    int gdzie2 = 0;

    for (int i = 0; i < n; i++)
    {
        t[i] = rand() % a + 1;
    }

    for (int i = 0; i < n; i++)
    {
        if (t[i]<x){
        x = t[i];
        }
        cout << t[i] << " ";
    }

    for (int i = 0; i < n; i++){

        if (t[i]==x){
            ile++;
        }
    }

    for(int i = 0;i < n-1 ; i++){
        if(t[L]==x){
            gdzie1=L;
        }else{
            L++;
        }
        if(t[P]==x){
            gdzie2=P;
        }else{
            P--;
        }
    }

    cout << endl <<"Najmniejsza wartosc: "<< x << endl;
    cout <<"Ile powtorzen: "<< ile << endl;
    cout <<"Najdalsza odleglosc: "<<gdzie2-gdzie1<<endl;

    system("pause");
}
```

7. Oszacowanie złożoności czasowej

Operacja dominująca:

i < N

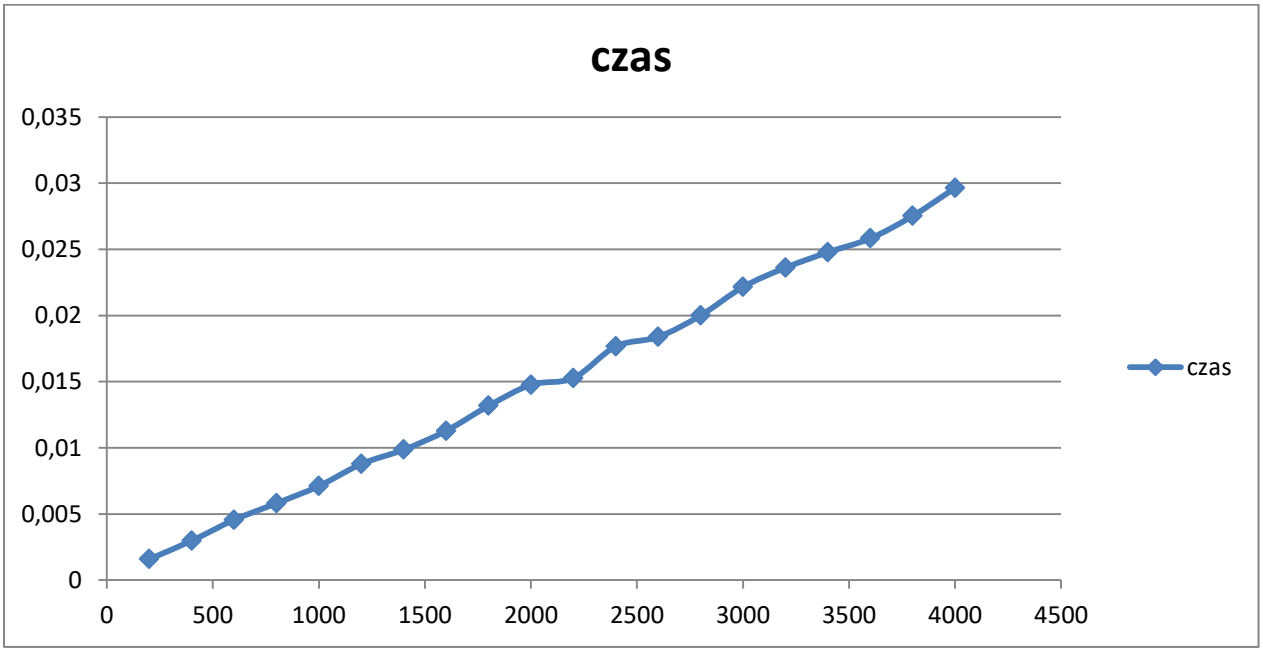
Funkcja złożoności (w najgorszym wypadku):

F(N) = N +N + N = N

Rząd złożoności w najgorszym przypadku:

F(N) = O(N) – złożoność liniowa

8. Wykres zależności czasu sortowania od rozmiaru zadania



Wniosek:

Wykres zależności czasu trwania algorytmu od liczby elementów tablicy potwierdza złożoność liniowa algorytmu (za wyjątkiem paru odchyleń)