Budowa i analiza algorytmów – ćwiczenia
Raport z realizacji mini-projektu

Numer projektu: (podać numer projektu)
Autor: (podać nazwisko i imię)
Numer albumu: (podać numer albumu)
Numer grupy zajęciowej: lZ01P0x
Termin oddania projektu: RRRR.MM.DD
Termin obrony: (podać numer terminu)

---

1. Treść zadania

(wpisać treść zadania)

tekst tekst tekst tekst

tekst tekst tekst tekst

tekst tekst tekst tekst

2. Opis słowny algorytmu

(Należy opisać w sposób ogólny zasadę działania naszego algorytmu, ewentualnie uzupełnić o zapis w pseudokodzie)

Dane wejściowe: lista N elementów, które można porównywać parami i określać właściwą kolejność występowania.

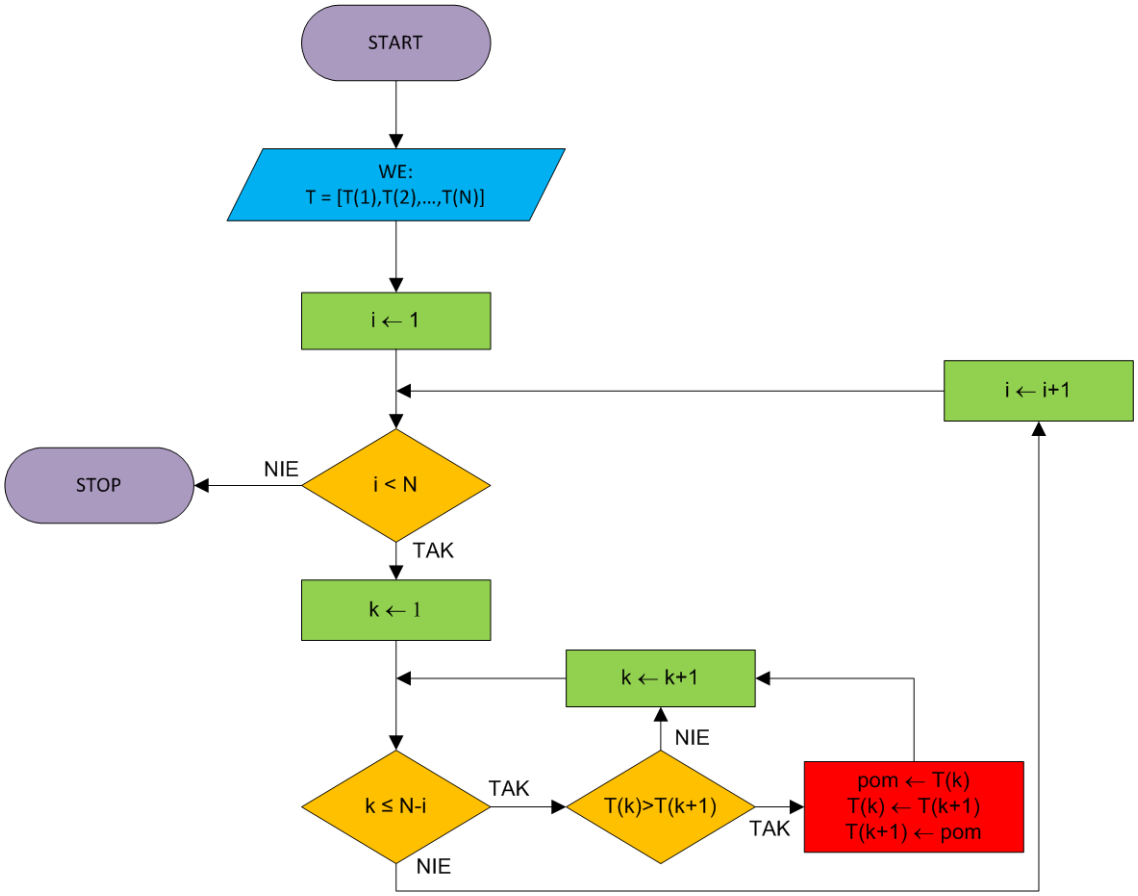
Oczekiwane dane wyjściowe: lista tych samych N elementów uporządkowana według zadanej kolejności.

1. wykonaj co następuje N – 1 razy:
1.1. wskaż pierwszy element listy,
1.2. wykonaj co następuje N – 1 razy:
1.2.1. porównaj wskazany element listy z następnym
1.2.2. jeśli te dwa elementy są w niewłaściwej kolejności, to zamień je miejscami,
1.2.3. wskaż następny element z listy.

tekst tekst tekst tekst

3. Schemat blokowy z omówieniem

(Należy umieścić schemat blokowy, ewentualnie dopisać objaśnienia, jeśli miałyby ułatwić analizę schematu)



Podpis rysunku

tekst tekst tekst tekst

tekst tekst tekst tekst

tekst tekst tekst tekst

4. Symulacja działania algorytmu dla przykładowych danych WE

T	N	i	i<N	k	k≤N-i	T(k)>T(i)	pom
[6, 4, 5]	3	1	1<3 TAK	1	1≤2 TAK	6>4 TAK	6
[4, 6, 5]				2	2≤2 TAK	6>5 TAK	6
[4, 5, 6]				3	3≤2 NIE		
		2	2<3 TAK	1	1≤1 TAK	4>5 NIE	
				2	2≤1 NIE		
		3	3<3 NIE				

5. Zapis algorytmu w języku SCILAB

```
// Funkcja do sortowania tablic metodą algorytmu bąbelkowego

function T=bubblesort(T)      // T - sortowana tablica
    N=length(T)               // N - liczba elementów tablicy T
    i=1
    while i<=N-1 do           // iteracja zewnętrzna
        k=1
        while k<=N-i do       // iteracja wewnętrzna
            if T(k)>T(k+1) then
                pom=T(k)
                T(k)=T(k+1)
                T(k+1)=pom
            end
            k=k+1
        end
        i=i+1
    end
endfunction
```

6. Oszacowanie złożoności czasowej

Operacja dominująca (elementarna):

porównanie $T(k) > T(k+1)$.

Funkcja złożoności (w najgorszym przypadku):

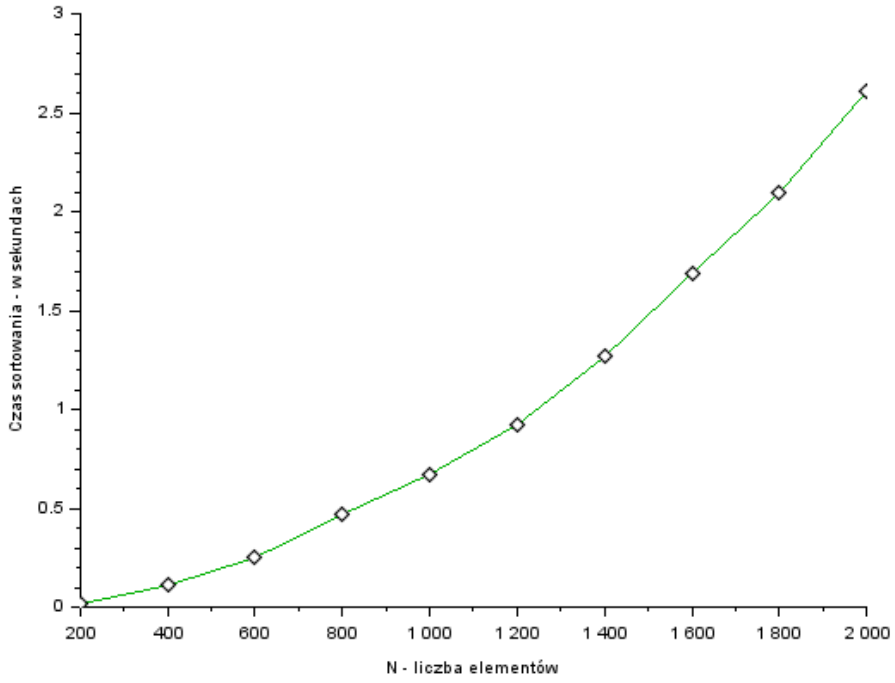
liczba porównań $T(k) > T(k+1)$ dla rozmiaru zadania N.

$$F(N) = (N-1) + (N-2) + \cdots + 2 + 1 = \frac{(N-1)+1}{2}(N-1) = \frac{N^2}{2} - \frac{N}{2}.$$

Rząd złożoności w najgorszym przypadku:

$$F(N) = O\left(\frac{N^2}{2} - \frac{N}{2}\right) = O\left(\frac{N^2}{2}\right) = O(N^2)$$ - złożoność kwadratowa.

7. Wykresu zależności czasu sortowania od rozmiaru zadania



(podpis wykresu)

Wniosek:

Wykres zależności czasu sortowania tablicy od liczby jej elementów potwierdza złożoność kwadratową algorytmu sortowania bąbelkowego.

8. Podsumowanie i wnioski

tekst tekst tekst tekst