Zadania II – Analiza Złożoności Algorytmów

Zadanie 1. *Jaką wartość zwraca funkcja MYSTERY. Przedstaw odpowiedź jako funkcję zależną od parametru n. Podaj również rząd wielkości rozwiązania używając notacji O.*

```
MYSTERY(n)
r ← 0
for i ← 1 to n − 1
    do for j ← i + 1 to n
           do for k ← 1 to j
                  do r ← r + 1
return r
```

Zadanie 2. *Niech $A[1..n]$ to tablica n-elementowa. Podaj złożoność pesymistyczną poniższego algorytmu ze względu na operacje porównania kluczy znajdujące się w liniach 4 oraz 6.*

```
MINMAX(A)
1  j ← 1
2  k ← 1
3  for i ← 2 to n
4      do if A[i] > A[j]
5             then j ← i
6             else  if A[i] < A[k]
7                      then k ← i
8  return (j, k)
```

Zadanie 3. *Podaj złożoność pesymistyczną i optymistyczną dla poniższego algorytmu sortującego tablicę $A[1..n]$. Operację porównania kluczy przyjmij jako dominującą.*

```
BUBBLESORT(A)
for i ← 1 to n
    do for j ← 2 to n
           do if A[j − 1] > A[j]
                 then swap(A[j − 1], A[j])
return A
```

Zadanie 4. *Poniżej zaprezentowano następujący algorytm:*

```
TAB(n)
pom ← 0
for i = 1 to n
    do for j = i to n
           do pom ← pom + 1
       pom ← pom − 1
return pom
```

Podaj:

(1) *dokładną liczbę jaką zwraca algorytm oraz*
(2) *oszacuj złożoność powyższego algorytmu ze względu na operację przypisania w liniach 1, 4 oraz 5.*

Zadanie 5. *Na wejściu znajduje się tablica A zawierająca n liczb rzeczywistych. Wyjście stanowi największa suma elementów dowolnego spójnego fragmentu tablicy z wejścia. Na przykład, jeżeli na wejściu znajduje się tablica*

$$[31, -41, 59, 26, -53, 58, 97, -93, -23, 84]$$

wówczas algorytm daje w wyniku sume elementów $A[3..7]$, czyli 187.
Poniżej zaprezentowano algorytm znajdujący taką sumę.

```
SUMA(A, n)
1  pom ← 0
2  for d = 1 to n
3     do suma ← 0
4        for j = d to n
5           do suma ← suma + A[j]
6              pom ← max(pom, suma)
7  return pom
```

Oszacuj złożoność powyższego algorytmu oraz podaj dokładną liczbę operacji przypisania w liniach 1, 3, 5 oraz 6.

Zadanie 6. *Jaką wartość zwraca funkcja TAB. Wyraź odpowiedź jako funkcję zależną od n oraz oszacuj rząd wielkości rozwiązania.*

```
TAB(n)
M ← 10
for i ← 1 to n − 1
   do for j ← i + 1 to n
         do for s ← 1 to j
               do M ← M + 1
return M
```

Zadanie 7. *Poniższy algorytm wyznacza x^n, gdzie $x \in \mathbb{R}$ oraz $n \in \mathbb{N}$.*

```
POTEGA(x, n)
p ← 1
while n > 0
   do if odd(n)
         then p ← p * x
              n ← n div 2
         x ← x * x
return p
```

Określ ile razy zostanie wykonane mnożenie (wiersz 4) w przypadku pesymistycznym. Funkcja odd(n) w (wiersz 3) oznacza kontrolę nieparzystości liczby tzn. zwraca TRUE gdy liczna jest nieparzysta.

Zadanie 8. *Poniższy algorytm wyznacza x^n, gdzie $x \in \mathbb{R}$ oraz $n \in \mathbb{N}$.*

```
POTEGA(x, n)
1  p ← 1
2  while n > 0
3     do p ← p * x
4        n ← n − 1
5  return p
```

Określ ile razy zostanie wykonane mnożenie (instrukcja w wierszu 3).

Zadanie 9. *Przeanalizuj pesymistyczny i optymistyczny czas działania poniższego algorytmu pod kątem operacji porównania kluczy oraz operacji przestawienia elementów w tablicy A n-elementowej.*

```
INSERT(A, n)
1  for i ← 2 to n
2     do x ← A[i]
3        j ← i − 1
4        while (j > 0) cand (x < A[j])
5           do A[j + 1] ← A[j]
6              j ← j − 1
7        A[j + 1] ← x
8  return A
```

Zadanie 10. *Przeanalizuj pesymistyczny i optymistyczny czas działania poniższego algorytmu pod kątem operacji porównania kluczy oraz operacji przestawienia elementów w tablicy A n-elementowej.*

```
SELECT(A, n)
 1  for i ← 1 to n − 1
 2     do k ← i
 3        x ← A[i]
 4        for j ← i + 1 to n
 5           do if A[j] < x
 6                 then k ← j
 7                      x ← A[j]
 8        A[k] ← A[i]
 9        A[i] ← x
10  return A
```

Zadanie 11. *Przeanalizuj czas działania poniższego algorytmu sortującego dla tablicy A n-elementowej. Tablica B to tablica wynikowa n-elementowa a tablica C to n-elementowa tablica pomocnicza.*

```
CountingSort(A, B, n, k)
for i ← 1 to n
    do inc(C[A[i]])
for i ← 2 to k
    do C[i] ← C[i] + C[i − 1]
for i ← n downto 1
    do B[C[A[i]]] ← A[i]
       dec(C[A[i]])
return B
```

Zadanie 12. *Dane są łańcuch* $S[1..n]$ *i wzorzec* $P[0..m-1]$*, gdzie* $1 \leqslant m \leqslant n$*. Poniższy algorytm wyznacza pozycję i występowania wzorca* P *w łańcuchu* S*, tzn.* $i = p$ *jeśli* $S[i..i+m-1] = P$*, a* $i = n - m + 1$ *jeśli wzorzec* P *nie jest podciągiem* S*.*

```
SZUKAJ(P, S, m, n)
 1  i ← 0
 2  znaleziono ← FALSE
 3  while (i ⩽ n − m) ∧ (¬znaleziono)
 4      do i ← i + 1
 5         r ← 0
 6         znaleziono ← TRUE
 7         while (r < m) ∧ (znaleziono)
 8             do znaleziono ← (P[r] == S[i + r])
 9                r ← r + 1
10  return i
```

Ile porównań (wiersz 8) wykonuje powyższy algorytm w przypadku pesymistycznym?

Zadanie 13. *Niech* Y *oraz* Z *będą macierzami* $n \times n$*.*
Poniższy algorytm przedstawia mnożenie dwóch macierzy.

```
MatrixMulti(Y, Z, n)
1  for i ← 1 to n
2      do for j ← 1 to n
3             do X[i, j] ← 0
4                for k ← 1 to n
5                    do X[i, j] ← X[i, j] + Y[i, k]· Z[k, j]
6  return X
```

Ile operacji + znajdującej się w linii 5 *zostanie wykonanych.*