Mateusz

1 laboratoria
Zadanie 1
Algorytm - skończony i uporządkowany zbiór instrukcji prowadzący do rozwiązania określonego problemu. Powinien być jednoznaczny, wykonany w skończonym czasie oraz poprawny.
Jak mierzyć i porównywać złożoność algorytmów? Używa się notacji asymptotycznych: * **O (O - bardzo górne ograniczenie)** – określa maksymalny czas działania algorytmu w najgorszym przypadku. * **Ω (Omega – dolne ograniczenie)** – określa minimalny czas działania w najlepszym przypadku. * **Θ (Theta – ścisła złożoność)** – gdy czas wykonania jest dokładnie ograniczony z góry i dołu przez ten sam wzór, gdy f = **O** i **Ω**

```
def selection_sort(arr):
n = len(arr)
for i in range(n):
min_idx = i
for j in range(i + 1, n):
if arr[j] < arr[min_idx]: # Znajdowanie najmniejszego elementu
min_idx = j
arr[i], arr[min_idx] = arr[min_idx], arr[i] # Zamiana miejscami
return arr
```

Liczba operacji: * wewnętrzna pętla wykonuje (n-1) + (n-2) + ... + 1 = **n(n-1)/2** porównań. * zamiany są wykonywane maksymalnie n razy.
Złożonosc: * **Najgorszy przypadek (O)**: O(n^2) * **Najlepszy przypadek (Ω)**: O(n^2) (algorytm zawsze wykonuje tyle samo porównań) * **Średni przypadek (Θ)**: O(n^2)

Zadanie 2

```
for(int i = n; i > 0; i--) { // Pętla 1
for(int j = 1; j < n; j++) { // Pętla 2
for(int k = 0; k < n; k += 2) { // Pętla 3
... // c operacji
}
}
}
```

Pętla 1 - n razy (od n do 1) Pętla 2 - (n-1) razy O(n) dla każdej wartosci i Pętla 3 - n/2 razy
Wewnątrz - c operacji

Łącznie = n*(n-1)*(n/2)*c Dla dużych: O(n^3)

Zadanie 3 T(N)=c*N^2 T(100)=1ms C=1/10000 T(5000)=(1/10000)*5000^2=2.5s

Zadanie 4

Wyrażenie	Czynnik dominujący	$O(n)$
$5 + 0.001n^3 + 0.025n$	n^3	$O(n^3)$
$500n + 100n^{1.5} + 50n\log_{10} n$	$n^{1.5}$	$O(n^{1.5})$
$0.3n + 5n^{1.5} + 2.5n^{1.75}$	$n^{1.75}$	$O(n^{1.75})$
$n^2 \log_2 n + n(\log_2 n)^2$	$n^2 \log_2 n$	$O(n^2 \log n)$
$n \log_3 n + n \log_2 n$	$n \log n$	$O(n \log n)$
$3 \log_8 n + \log_2 \log_2 \log_2 n$	$\log n$	$O(\log n)$
$100n + 0.01n^2$	n^2	$O(n^2)$
$0.01n + 100n^2$	n^2	$O(n^2)$
$2n + n^{0.5} + 0.5n^{1.25}$	$n^{1.25}$	$O(n^{1.25})$
$0.01n \log_2 n + n(\log 2n)^2$	$n(\log n)^2$	$O(n(\log n)^2)$
$100n \log_3 n + n^3 + 100n$	n^3	$O(n^3)$
$0.003 \log_4 n + \log_2 \log_2 n$	$\log n$	$O(\log n)$
$2^n + n^{10}$	2^n	$O(2^n)$
$5^n + n! + n^{20}$	5^n	$O(5^n)$

1. O(1)
2. O(logn)
3. O(n^c) dla c(0,1)
4. O(n)
5. O(nlogn)
6. O(n^k)
7. O(n!)
8. O(c^n)
9. O(n^n)

Zadanie 5

Zadanie 5

Poniżej przedstawiono własności notacji asymptotycznej dla funkcji $f \equiv f(n)$ i $g \equiv g(n)$. Zidentyfikuj poprawne stwierdzenia. Spróbuj poprawić błędne.

Stwierdzenie	Prawda/Fałsz	Proponowana poprawka
$O(f+g) = O(f) + O(g)$	F	$O(max(f,g))$
$O(f \cdot g) = O(f) \cdot O(g)$	T	
$5n + 8n^2 + 100n^3 \in O(n^4)$	T	
$100n^5 \in O(n!)$	T	

Zadanie 6 1 funkcja: O(n) 2 funkcja: O(n^2) 3 funkcja: O(1) 4 funkcja: O(n!)

Zadanie 7 T(n)<=cn^3 a0 <= |a0|n^3 a1n<=|a1|n^3 a2n^2<=|a2|n^3 a3n^3<=|a3|n^3 |T(n)|<=(|a0|+|a1|+|a2|+|a3|)n^3 wybieramy c=|a0|+|a1|+|a2|+|a3|, z definicji notaci dużego-O wynika że T(n) należy do O(n^3)

2 laboratoria
Zadanie 1

```
def merge_sort_pom(array, left, right, mid):
left_half = array[left:mid+1]
right_half = array[mid+1:right+1]

i = j = 0
k = left

while i < len(left_half) and j < len(right_half):
if left_half[i] < right_half[j]:
array[k] = left_half[i]
i += 1
else:
array[k] = right_half[j]
j += 1
k += 1

while i < len(left_half):
array[k] = left_half[i]
i += 1
k += 1

while j < len(right_half):
```

```
array[k] = right_half[j]
j += 1
k += 1

def merge_sort(array, left, right):
if left >= right:
return
mid = (left+right) // 2
merge_sort(array, left, mid)
merge_sort(array, mid+1, right)
merge_sort_pom(array, left, right, mid)
```

Zadanie 2-4

Na platformie Hackerrank

Zadanie 4

Zakładamy wysokosc h liczoną od 0 (kopiec składający się tylko z korzenia ma wysokosc 0)

maksymalna liczba elementów: 2^(h+1) - 1

minimalna liczba elementów: 2^h

Zadanie 5

W pełnym drzewie binarnym poziom h zawiera maks. 2^h węzłów

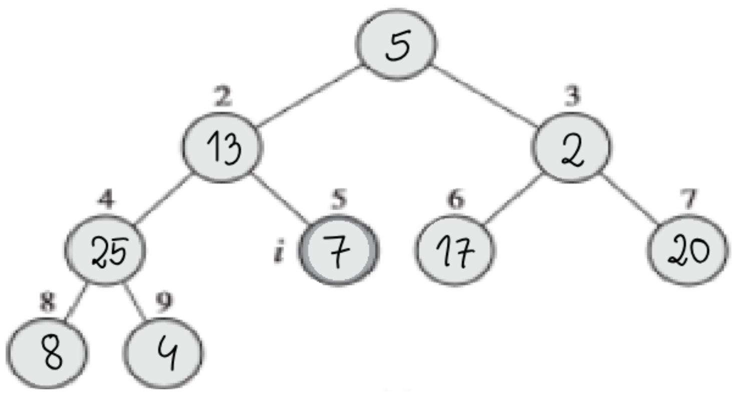
Maksymalna liczba elementów w kopcu o wysokosci h: 2^(h+1)-1, natomiast minimalna: 2^h

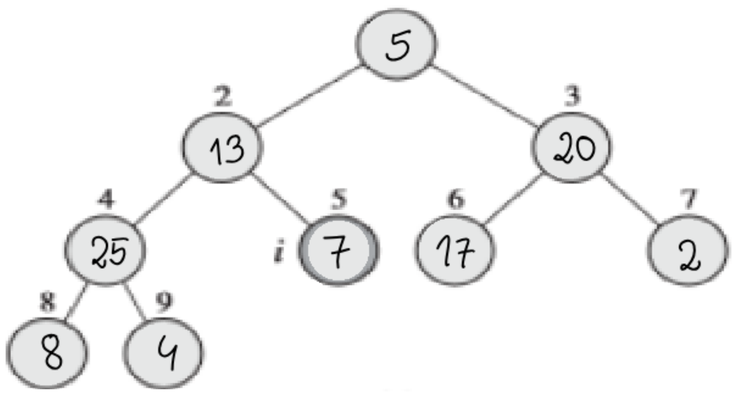
Załóżmy że n jest dowolną liczbą węzłów kopca wysokosci h: 2^h <= n <= 2^(h+1) - 1

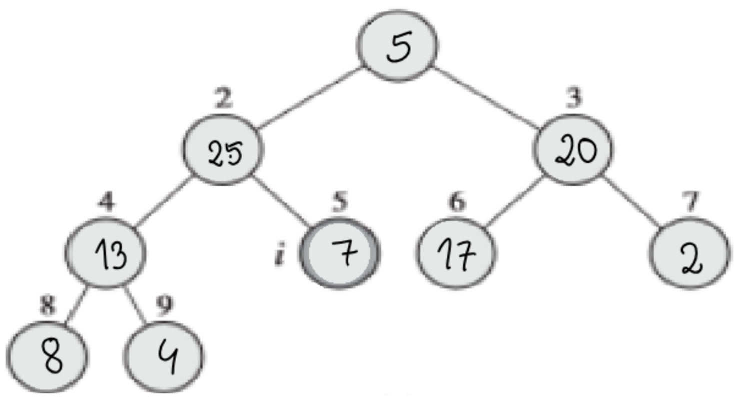
Biorąc logarytm dwójkowy po obu stronach nierownosci: h <= lg n <= h+1

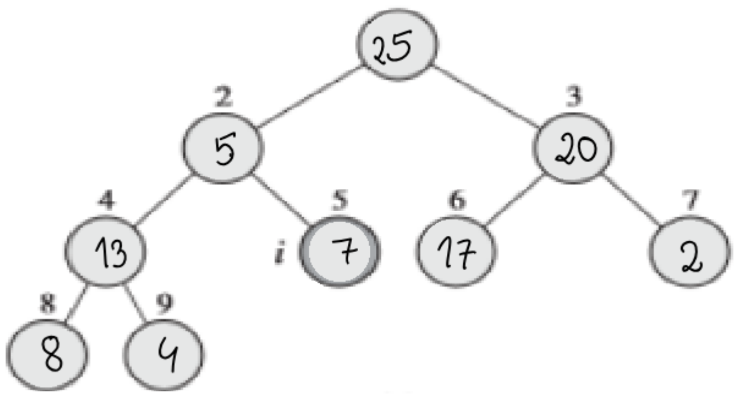
Po zaokrągleniu w dół: h = ⌊lgn⌋

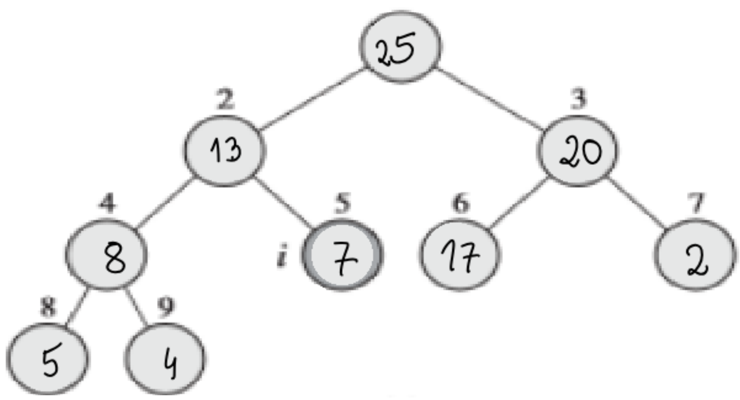
Zadanie 6


5
2 13
3 2
4 25
5 i 7
6 17
7 20
8 8
9 4


5
2 13
3 20
4 25
5 i 7
6 17
7 2
8 8
9 4


5
2 25
3 20
4 13
5 i 7
6 17
7 2
8 8
9 4





3 laboratoria

Zadanie 1

a) $T(n)=T(n-1)+1$

$T(n-1)=T(n-2)+1$

$T(n-2)=T(n-3)+1$

ogólnie: $T(n)=T(n-k)+k$

podstawiamy $k=n-1$

$T(n)=T(1)+(n-1)$

ponieważ $T(1)=1$: $T(n)=1+n-1=$**n**

b) $T(n)=2T(n/2)+n$

$T(n/2)=2T(n/4)+n/2$

$T(n/4)=2T(n/8)+n/4$

$T(n)=2(2T(n/4)+n/2)+n=4T(n/4)+n+n=4T(n/4)+2n=4(2T(n/8)+n/4)+2n=8T(n/8)+3n$

$T(n)=2^k * T(n/2^k) + kn$

rekurencja konczy się gdy $n/2^k = 1$ czyli $k=\log_2 n$, wówczas $T(1)=1$

$T(n)=2^{(\log_2 n)} * T(1) + (\log_2 n)n$

$T(n)=O(n * \log_2 n)$

c) $T(n)=T(n^{(1/2)})+1$

$T(n^{(1/2)})=T(n^{(1/4)})+1$

$T(n^{(1/4)})=T(n^{(1/8)})+1$

$T(n)=T(n^{(1/4)})+2$

T(n)=T(n^(1/8))+3
T(n)=T(n^(1/2^k))+k
rekurencja konczy się gdy n^(1/2^k)=1 czyli