3. Theoretical Part

3.1. Big-O notation

1. $10n \log n + 500n + n^2 + 123 = O(n^2)$

   Proof. By definition of big-O notation, it is sufficient to show that there exist constants $c > 0$ and $n_0 > 0$ such that for all $n \geq n_0$ we have $n \log n + n + n^2 \leq c \cdot n^2$ (constants can be omitted, because they do not affect the growth of the function).

   Thus, let $n_0 = 0$ and $c = 2$. Then, for $n \geq n_0$ we have
   $$n \log n + n + n^2 \leq 2n^2$$

2. $n^{\frac{9}{2}} + 7n^4 \log n + n^2 = O(n^{\frac{9}{2}})$

   Proof. By definition of big-O notation, it is sufficient to show that there exist constants $c > 0$ and $n_0 > 0$ such that for all $n \geq n_0$ we have $n^{\frac{9}{2}} + n^4 \log n + n^2 \leq c \cdot n^{\frac{9}{2}}$ (constants can be omitted, because they do not affect the growth of the function).

   Thus, let $n_0 = 1$ and $c = 2$. Then, for $n \geq n_0$ we have
   $$n^{\frac{9}{2}} + n^4 \log n + n^2 \leq 2n^{\frac{9}{2}}$$

3. $6^{n+1} + 6(n+1)! + 24n^{42} = O((n+1)!)$

   Proof. By definition of big-O notation, it is sufficient to show that there exist constants $c > 0$ and $n_0 > 0$ such that for all $n \geq n_0$ we have
   $6^{n+1} + (n+1)! + n^{42} \leq c \cdot (n+1)!$ (constants can be omitted, because they do not affect the growth of the function).

   Thus, let $c = 10$. Then, since the growth of the factorial exceeds the growth of any degree we have
   $$6^{n+1} + (n+1)! + n^{42} \leq 10 \cdot (n+1)!$$

3.2. Dynamic binary search

1. Search

Instructions	Cost	Times
for array in arrays {	c_1	a
int l = 0, r = array.length;	c_2	a * 2
while (l < r) {	c_3	a * log(n)
int mid = (l + r) / 2;	c_4	a * log(n)
if (array[mid] < value) { l = mid + 1; }	c_5	a * 2 * log(n)
else if (array[mid] > value) { r = mid; }	c_6	a * 2 * log(n)
else { return true; }	c_7	a * log(n)
}	0	a * 2
}	0	1
return false;	c_8	1

$T(n) = 5a + 7a * log(n) + 2 = O(a * log(n))$
Asymptotic complexity analysis:
$O(log(n) * log(n + 1)) = O(log^2(n))$

2. Insert

Instructions	Cost	Times
function insert(value) {		
values = new array of size 1;	c_1	1
values[0] = value;	c_2	1
insertMany(values);	c_3	$O(k^k)$

}		
function insertMany(values) {		
if (arrays is empty) { arrays.add(values); }	c_4	2
else {		
head = arrays[0];	c_5	1
if (arrays.head.size > values.size) {	c_6	1
arrays.add(values)	c_7	1
} else {		
merged = new array of size (values.size + head.size);	c_8	1
i = 0; j = 0;	c_9	2
for (k from 0 to merged.size - 1) {	c_{10}	$O(k)$
if (j >= head.size) { merged[k] = values[i++];	c_{11}	$2 * O(k)$
} else if (i >= values.size) { merged[k] = head[j++];	c_{12}	$2 * O(k)$
} else if (values[i] <= head[j]) { merged[k] = values[i++];	c_{13}	$2 * O(k)$
} else { merged[k] = head[j++]; }	c_{14}	$O(k)$
}		
arrays.remove(0);	c_{15}	1
insertMany(merged);	c_{16}	1
}		
}		
}		

Running time: $O(k^{k+1})$

Asymptotic complexity: $O(k^{k})$

3.3. Recurrences and Master Theorem

$T(n) = \sqrt{k} \cdot T(\frac{n}{k^2}) + c \cdot \sqrt[3]{n}$

$T(1) = 0$

$a = \sqrt{k}$

$b = k^2$

$f(n) = c \cdot \sqrt[3]{n}$

$log_{k^2}\sqrt{k} = \frac{1}{4} = c\ (critical)$

$c_p = \frac{1}{3}$

$\frac{1}{3} > \frac{1}{4}$ => Third case

Regularity condition:

$\sqrt{k} \cdot c\sqrt[3]{\frac{n}{k^2}} \leq l \cdot c \cdot \sqrt[3]{n}$

$\sqrt{k} \cdot \sqrt[3]{\frac{1}{k^2}} \leq l$

$\sqrt[6]{\frac{1}{k}} \leq l$

If $k \geq 1$, then $l < 1$, so the regularity condition is not violated.

1. $T(n) = \theta(\sqrt[3]{n})$
2. $3^{rd}\ case$. Because if $k \geq 1$, the regularity condition is not violated and $c > log_b a$.