1 Asymptotic computational complexity

Task 1. Расположите функции по увеличению скорости роста
• $f(n) = \log n$ • $f(n) = 3$ • $f(n) = n^3$ • $f(n) = n^2 * \log n$ • $f(n) = 2^n * \log n$ • $f(n) = 2^n$ • $f(n) = n * (\log n)^2$

Task 2. Поставьте в пары функции с одинаковой скоростью роста.

1. $f(n) = 4$
2. $f(n) = 4 * n + 23$
3. $f(n) = (\log n)^2 + \log n$
4. $f(n) = (n^2 + n)^3$
5. $f(n) = n * (n^2 + 1) * (n^2 + 2)$

(a) $g(x) = n + 1$

(b) $g(x) = e^2$

(c) $g(x) = n^5 + 4 * n^2$

(d) $g(x) = \log n^2 + (\log n^2)^2$

(e) $g(x) = n^6 + 4$

Task 3. Представьте, что у нас есть два алгоритма, которые решают одну и ту же задачу. Количество операций, совершаемых первым алгоритмом, равно $f(n) = 5 \cdot n \cdot \log_{10} n$, а вторым — $g(n) = 25 \cdot n$. Несмотря на то, что мы знаем, что скорость роста $f(n)$ больше, чем $g(n)$ — легко убедиться, что при небольших n нам следует выбрать $f(n)$. А при каком $n > 0$ количество операций, выполняемых данными алгоритмами, одинаково?

Task 4. Выберите верные утверждения из списка

1. $5n + 8 * n^2 + 100 * n^3 = O(n^4)$
2. $O(f + g) = O(f) + O(g)$
3. $5n + 8 * n^2 + 100 * n^3 = O(n * \log n)$
4. $O(f * g) = O(f) * O(g)$
5. If $f = O(g)$ and $f = O(h)$ then $g = O(h)$

Task 5. Выберите верные утверждения из списка

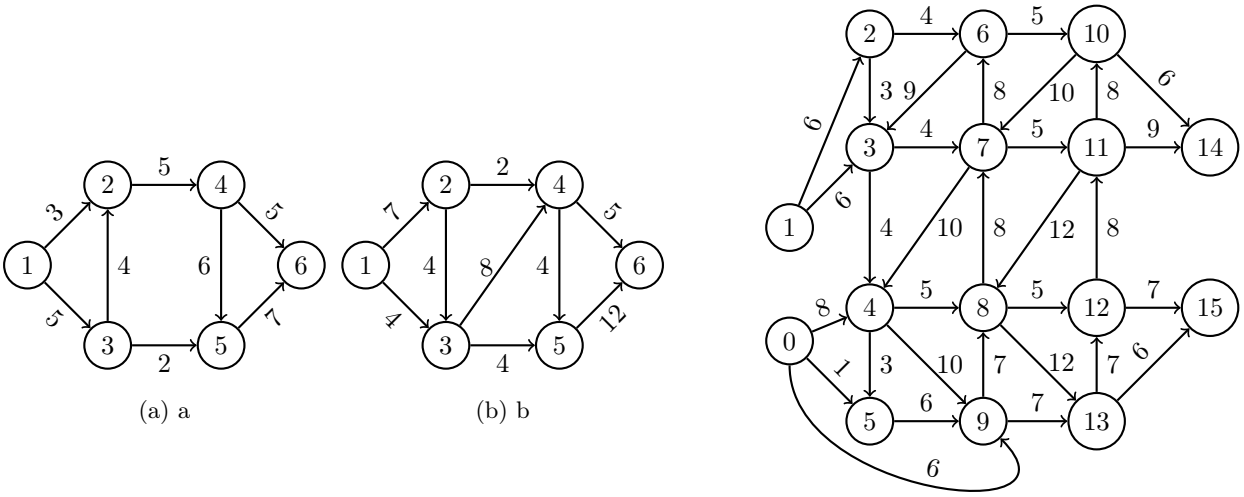
1. $100 * n * \log n + n^3 + 100 * n = O(n^3)$
2. $2^n = O(n^{100500})$
3. $500 * n + 100 * n^{1.5} + 50 * n * \log n = O(n * \log n)$
4. $5 + 0.001 * n^3 + 0.025 * n = O(n^3)$
5. $100 * n * \log n + 0.1 * n^2 = O(n^2)$
6. $100 * n * \log n + n^3 + 100 * n = O(n^4)$
7. $0.003 * \log n + \log \log n = O(\log \log n)$

Task 6. Подсчитать асимптотическую сложность алгоритмов:

1. Bubble sort
2. Merge sort
3. Binary Tree
4. Sorted and unsorted linked list
5. Graph via adjacency list with operations memory, find_max, add_edge, remove_edge, add_vertex, remove_vertex, neighborhood_check.

Task* 7. Подсчитать асимптотическую сложность алгоритмов:

1. Quick sort
2. Heap sort
3. Prim's algorithm



(a) a (b) b

(c) c

Figure 1: graphs

4. B-tree
5. AVL tree
6. binomial tree
7. Bfs, dfs
8. Iterative matrix multiplication and Strassen algorithm
9. Graph (other representations) list with operations memory, find_max, add_edge, remove_edge, add_vertex, remove_vertex, neighborhood_check.

Task* 8. Докажите, что:

1. $n^2 + 100 * n + 88 = \Theta(n^2)$
2. $\lfloor n + \frac{1}{2} \rfloor = \Theta(n)$
3. $|n + \frac{1}{2}| = \Theta(n)$
4. $2(n-1) + \frac{n(n+1)}{2} + 4\frac{n(n-1)}{2} = \Theta(n^2)$
5. $\sum_{k=1}^{n}(k+2) = \Theta(n^2)$

Task* 9. Расположите приведенные ниже функции по скорости их ассимптотического роста, т.е. постройте такую последовательность функций $g_1, g_2, \ldots, g_{30}$, что $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots, g_{29} = \Omega(g_{30})$. Расбейте полученный список на классы эквивалентности так, что f и g принадлежать одному классу $\Leftrightarrow f(n) = \Theta(g(n))$.

$\lg \lg^* n$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n * 2^n$	$n^{\lg \lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^* \ln n$	$2^{\sqrt{2*\lg n}}$	n	2^n	$n * \lg n$	$2^{2^{n+1}}$

2 Flows

Task 10. Решить задачу нахождения максимального потока в транспортной сети с помощью алгоритма Форда–Фалкерсона, и построить разрез сети. 0 — исток, 7 — сток. Дуги:

$$
\begin{array}{lllll}
m[0,1]=39 & m[4,7]=44 & m[6,3]=33 & m[5,7]=53 & m[0,2]=10 \\
m[4,2]=18 & m[6,7]=95 & m[5,4]=16 & m[0,3]=23 & m[2,5]=61 \\
m[2,1]=81 & m[6,5]=71 & m[1,4]=25 & m[2,6]=15 & m[3,2]=20
\end{array}
$$

Task 11. Решить задачу нахождения максимального потока в транспортной сети. 1. fig 1a; 2. fig 1b; 3^*. fig 1c.
Task 12 [Programming]. Реализовать алгоритмы:

- Форда–Фалкерсона
- Эдмондса–Карпа
- Диница
- Масштибирование потока