1 Глава 1

1.1 Поиск контрпримеров

1.1.1 #1

Условие. Докажите, что значение $a+b$ может быть меньшим, чем значение $\min(a, b)$.

Допустим:

$$a = 0, b = -1 \tag{1}$$

Тогда:

$$0 + (-1) = \quad -1 \tag{2}$$

$$-1 \quad < 0 = \min(0, 1) \tag{3}$$

Что и требовалось доказать.

1.1.2 #2

Условие. Докажите, что значение $a \times b$ может быть меньшим, чем значение $\min(a, b)$.

Допустим:

$$a = \frac{1}{2}, b = \frac{1}{4} \tag{4}$$

Тогда:

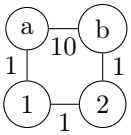
$$\frac{1}{2} \times \frac{1}{4} = \frac{1}{8} \tag{5}$$

$$\frac{1}{8} < \min(\tfrac{1}{2}, \tfrac{1}{4}) = \tfrac{1}{4} \tag{6}$$

Что и требовалось доказать.

1.1.3 #3

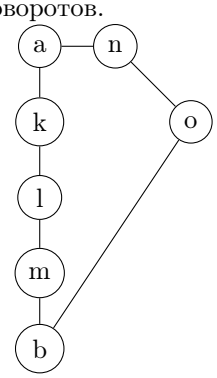
Условие. Начертите сеть дорог с двумя точками a и b, такими, что маршрут между ними, преодолеваемый за кратчайшее время, не является самым коротким.



Цена маршрута «a,1,2,b» равна 3, а цена маршрута «a,b» равна 10. То есть, маршрут, проходящий через больше точек, более длинный, преодолевается за меньшее время. Что и требовалось доказать.

1.1.4 #4

Условие. Начертите сеть дорог с двумя точками a и b, самый короткий маршрут между которыми не является маршрутом с наименьшим числом поворотов.



В данной сети дорог кратчайший маршрут от a до b это $anob$, в котором 3 ребра, но два поворота. Однако, маршрут без поворотов вообще, $aklmb$ не самый короткий — в нём 4 ребра. Что и требовалось доказать.

1.1.5 #5

Условие. Задача о рюкзаке: имея множество целых чисел $S = \{s_1, s_2, \ldots, s_n\}$ и целевое число T, найти такое подмножество множества S, сумма которого в точности равна T. Например, множество $S = \{1, 2, 5, 9, 10\}$, содержит такое подмножество, сумма элементов которого равна T = 22, но не T = 23.

Найти контрпримеры для каждого из следующих алгоритмов решения задачи о рюкзаке, т. е., нужно найти такое множество S и число T, при которых подмножество, выбранное с помощью данного алгоритма, не до конца заполняет рюкзак, хотя правильное решение и существует:

- вкладывать элементы множества S в рюкзак в порядке слева направо, если они подходят (т. е., алгоритм «первый подходящий»);
- вкладывать элементы множества S в рюкзак в порядке от наименьшего до наибольшего (т. е., используя алгоритм «первый лучший»);
- вкладывать элементы множества S в рюкзак в порядке от наибольшего до наименьшего.

Первый подходящий. $T = 10, S = \{9, 2, 2, 2, 2, 2\}$
Первый лучший. $T = 6, S = \{1, 2, 5, 4\}$
От самого большого. $T = 6, S = \{3, 2, 5, 4\}$

1.1.6 #6

Условие. Задача о покрытии множества: имея семейство подмножеств $S_1, \ldots, S_m$ универсального множества $U = \{1, \ldots, n\}$, найдите семейство подмножеств

$T \subset S$ наименьшей мощности, чтобы $\bigcup_{t_i} \in T^{t_i} = U$. Например, для семейства подмножеств $S_1 = \{1, 3, 5\}$, $S_2 = \{2, 4\}$, $S_3 = \{1, 4\}$, $S_5 = \{2, 5\}$ покрытием множества будет семейство подмножеств S_1 и S_2.

Приведите контрпример для следующего алгоритма: выбираем самое мощное подмножество для покрытия, после чего удаляем все его элементы из универсального множества; повторяем добавление подмножества, содержащего наибольшее количество неохваченных элементов, пока все элементы не будут покрыты.

$$S_1 = \{1, 3, 5, 7, 9\} \tag{7}$$
$$S_2 = \{1, 2\} \tag{8}$$
$$S_3 = \{3, 4\} \tag{9}$$
$$S_4 = \{5, 6\} \tag{10}$$
$$S_5 = \{7, 8\} \tag{11}$$
$$S_6 = \{9, 10\} \tag{12}$$

Наилучшее покрытие это $S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6$, но жадный алгоритм вначале выберет S_1 как самое мощное подмножество, а потом не останется ничего другого, как добавлять все остальные подмножества, то есть, в итоге алгоритм выберет все исходные подмножества в качестве результата работы, $T = S$.

1.2 Доказательство правильности

1.2.1 #7

Условие. Докажите правильность следующего рекурсивного алгоритма умножения двух натуральных чисел для всех целочисленных констант $c \geqslant 2$:

```
1: function MULTIPLY(y, z)
2:     if z = 0 then
3:         return 0
4:     else
5:         return multiply(cy, ⌊z/c⌋) + y * (z mod c)
6:     end if
7: end function
```

Вычислим результат работы multiply на каком-нибудь простом примере:

$$
\begin{aligned}
& c = 2, y = 2, z = 3 && (13)\\
& multiply(2, 3) && (14)\\
= \; & multiply(2 * 2, \lfloor 3/2 \rfloor) + 2 * (3 \mod 2) && (15)\\
= \; & multiply(4, 1) + 2 * 1 && (16)\\
= \; & multiply(4, 1) + 2 && (17)\\
= \; & multiply(4 * 2, \lfloor 1/2 \rfloor) + 4 * (1 \mod 2) + 2 && (18)\\
= \; & multiply(8, 0) + 4 * 1 + 2 && (19)\\
= \; & 0 + 4 + 2 && (20)\\
= \; & 6 && (21)
\end{aligned}
$$

Пусть $z = 0$. Тогда $multiply(y, z) = 0$.

Допустим теперь, что алгоритм правильный, то есть $multiply(y, z) = yz$ при:

$$
\begin{aligned}
z &\leqslant n && (22)\\
y &\geqslant 1 && (23)\\
c &\geqslant 2 && (24)
\end{aligned}
$$

Теперь докажем, что $multiply(y, n + 1) = y(n + 1)$.

$$
\begin{aligned}
& mutiply(y, n + 1) = && (25)\\
& multiply(y * c, \lfloor \frac{n+1}{c} \rfloor) + y * ((n + 1) \mod c) = && (26)\\
& y * c * \lfloor \frac{n+1}{c} \rfloor + y * ((n + 1) \mod c) = && (27)\\
& y * (c * \lfloor \frac{n+1}{c} \rfloor + (n + 1) \mod c) && (28)
\end{aligned}
$$

Теперь докажем, что $c * \lfloor \frac{n+1}{c} \rfloor + (n + 1) \mod c) = n + 1$.

$$
\begin{aligned}
n + 1 &\equiv z && (29)\\
c\lfloor \frac{z}{c} \rfloor + z \mod c &= z && (30)
\end{aligned}
$$

z/c в случае деления произвольных натуральных чисел — операция с остатком. $\lfloor z/c \rfloor$ же избавляется от этого остатка.

В то же время, $z \mod c$ это операция получения *остатка от деления* z на c.

Формула (30) может использоваться как есть, потому что следует из определения операций, которые в ней используются.

Если мы умножим c на операцию, убирающую остаток от деления z на c, то получим как бы «восстановленную» z без остатка после деления.

$$3 \times \lfloor \frac{10}{3} \rfloor = 3 \times 3 = 9 \tag{31}$$

Но если мы потом прибавим буквально остаток от этого же деления, мы получим исходное число!

$$10 \mod 3 = 1 \tag{32}$$

А $9 + 1 = 10$, как и следует из объяснения выше.
Таким образом, из (30) и (28) следует:

$$y \times (c \times \lfloor \frac{n+1}{c} \rfloor + (n+1) \mod c) = y \times (n+1) \tag{33}$$

$$\Rightarrow multiply(y, n+1) = y \times (n+1) \tag{34}$$

Что и требовалось доказать.

1.2.2 #8

Условие. Докажите правильность следующего алгоритма вычисления полинома $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x^+ a_0$:

```
1: function HORNER(A, x)
2:     p = A_n
3:     for i ← n − 1, 0 do
4:         p = p × x + A_i
5:     end for
6:     return p
7: end function
```

Пусть $n = 0$. Тогда цикл не выполнится ни разу и результатом будет A_0.
Пусть $n = 1$. Тогда:

$$p = A_1 \tag{35}$$

$$i = 0: \quad p = A_1 \times x + A_0 \tag{36}$$

Пусть $n = 2$. Тогда:

$$p = A_2 \tag{37}$$

$$i = 1: \quad p = A_2 \times x + A_1 \tag{38}$$

$$i = 0: \quad p = (A_2 \times x + A_1) \times x + A_0 = A_2 \times x^2 + A_1 \times x + A_0 \tag{39}$$

Допустим, что алгоритм правильный при любых n. Посмотрим, что происходит при $n + 1$:

$$p = A_{n+1} \tag{40}$$
$$i = n: \quad p = A_{n+1} \times x + A_n \tag{41}$$
$$i = n-1: \quad p = (A_{n+1} \times x + A_n) \times x + A_{n-1} \tag{42}$$
$$\ldots \tag{43}$$
$$i = n - n = 0: \quad p = A_{n+1}x^{n+1} + A_n x^n + A_{n-1}x^{n-1} + \ldots + A_{n-n} \tag{44}$$

Что и требовалось доказать.

1.2.3 #9

Условие. Докажите правильность следующего алгоритма сортировки:

```
function bubblesort(A: list[1...n])
    var int i,j
    for i from n to 1
        for j from 1 to i - 1
            if (A[j] > A[j+1])
                swap A[j] and A[j + 1]
```

Пусть $n = 2$. Тогда:

$$i = 2, j = 1: \quad A[1] > A[2]?swap : keep \tag{45}$$

Пусть алгоритм может отсортировать любой массив длиной n. Может ли он отсортировать массив длиной $n + 1$?

Передача в этот алгоритм массив длиной $n+1$ эквивалентна замене его на следующий код:

```
function bubblesort(A: list[1...n+1])
    var int i,j
    for j from 1 to n + 1 - 1
        if (A[j] > A[j + 1])
            swap A[j] and A[j + 1]
    for i from n to 1
        for j from 1 to i - 1
            if (A[j] > A[j+1])
                swap A[j] and A[j + 1]
```

По индукции мы допустили, что основная пара вложенных циклов действительно отсортирует все элементы от 1 до n.

Мы же добавили в код ещё один цикл:

```
for j from 1 to n
    if (A[j] > A[j + 1])
        swap A[j] and A[j + 1]
```

Этот цикл проверит два новых элемента: `A[n]` и `A[n+1]`, последние в списке. Элемент `A[n+1]` идёт после отсортированных элементов `A[1] ... A[n]`. Если они не в правильном порядке, он их поменяет местами, в точности как случай когда $n = 2$.

Таким образом, этот алгоритм действительно сортирует массивы любой длины.

1.3 Математическая индукция

1.3.1 #10

Условие. Докажите, что:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{46}$$

для $n \geqslant 0$.

По индукции, пусть $n = 1$:

$$1 = \frac{1 \times (1+1)}{2} \tag{47}$$

$$1 = \frac{1 \times 2}{2} \tag{48}$$

$$1 = 1 \tag{49}$$

Пусть верно для n. Вычислим для $n + 1$:

$$1 + 2 + \ldots + n + (n+1) = \frac{(n+1)(n+2)}{2} \tag{50}$$

$$\frac{n(n+1)}{2} + n + 1 = \frac{(n+1)(n+2)}{2} \tag{51}$$

$$n(n+1) + 2n + 2 = (n+1)(n+2) \tag{52}$$

$$n^2 + n + 2n + 2 = n^2 + n + 2n + 2 \tag{53}$$

Что и требовалось доказать.

1.3.2 #11

Условие. Докажите, что:

$$\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6, n \geqslant 0 \tag{54}$$

Пусть $n = 1$, тогда:

$$1^2 = \frac{1 \times (1+1)(2 \times 1 + 1)}{6} \quad (55)$$

$$1 = \frac{6}{6} \quad (56)$$

$$1 = 1 \quad (57)$$

Теперь пусть формула верна для любых n. Вычислим для $n+1$:

$$\sum_{i=1}^{n} i^2 + (n+1)^2 = \frac{(n+1)(n+1+1)(2 \times (n+1)+1)}{6} \quad (58)$$

$$\frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{(n+1)(n+2)(2n+3)}{6} \quad (59)$$

$$n(n+1)(2n+1) + 6n^2 + 12n + 6 = (n+1)(n+2)(2n+3) \quad (60)$$

$$(n^2+n)(2n+1) + 6n^2 + 12n + 6 = (n^2+n+2n+2)(2n+3) \quad (61)$$

$$2n^3 + 2n^2 + n^2 + n + 6n^2 + 12n + 6 = 2n^3 + 2n^2 + 4n^2 + 4n + 3n^2 + 3n + 6n + 6 \quad (62)$$

$$2n^3 + 9n^2 + 13n + 6 = 2n^3 + 9n^2 + 13n + 6 \quad (63)$$

Что и требовалось доказать.

1.3.3 #12

Условие. Докажите, что:

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}, n \geqslant 0 \quad (64)$$

Пусть $n = 1$.

$$1 = \frac{1 \times 2^2}{4} \quad (65)$$

$$1 = 4/4 \quad (66)$$

$$1 = 1 \quad (67)$$

Теперь пусть формула верна для любых n. Вычислим для $n+1$:

$$\sum_{i=1}^{n} i^3 + (n+1)^3 = \frac{(n+1)^2(n+2)^2}{4} \quad (68)$$

$$\frac{n^2(n+1)^2}{4} + (n+1)^3 = \frac{(n+1)^2(n+2)^2}{4} \quad (69)$$

$$n^2(n+1)^2 + 4(n+1)(n+1)^2 = (n+1)^2(n+2)^2 \quad (70)$$

$$(n+1)^2(n^2+4n+4) = (n+1)^2(n+2)^2 \quad (71)$$

$$n^2 + 4n + 4 = (n+2)^2 \quad (72)$$

$$n^2 + 4n + 4 = n^2 + 4n + 4 \quad (73)$$

Что и требовалось доказать.

1.3.4 #13

Условие. Докажите, что:

$$\sum_{i=1}^{n} i(i+1)(i+2) = \frac{n(n+1)(n+2)(n+3)}{4}, n \geqslant 0 \tag{74}$$

Пусть $n = 1$.

$$2 \times 3 = \frac{2 \times 3 \times 4}{4} \tag{75}$$
$$6 = 6 \tag{76}$$

Теперь пусть формула верна для любых n. Вычислим для $n+1$:

$$\sum_{i=1}^{n} i(i+1)(i+2) + (n+1)(n+2)(n+3) = \frac{(n+1)(n+2)(n+3)(n+4)}{4} \tag{77}$$
$$\frac{n(n+1)(n+2)(n+3)}{4} + (n+1)(n+2)(n+3) = \frac{(n+1)(n+2)(n+3)(n+4)}{4} \tag{78}$$
$$n(n+1)(n+2)(n+3) + 4(n+1)(n+2)(n+3) = (n+1)(n+2)(n+3)(n+4) \tag{79}$$
$$n+4 = n+4 \tag{80}$$

Что и требовалось доказать.

1.3.5 #14

Условие. Докажите, что:

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}, n \geqslant 1, a \neq 1 \tag{81}$$

Пусть $n = 1$. Тогда:

$$a^0 + a^1 = \frac{a^2-1}{a-1} \tag{82}$$
$$(a-1)(a+1) = a^2 - 1 \tag{83}$$
$$a^2 - a + a - 1 = a^2 - 1 \tag{84}$$
$$a^2 - 1 = a^2 - 1 \tag{85}$$

Теперь пусть формула верна для любых n. Вычислим для $n+1$:

$$\sum_{i=0}^{n} a^i + a^{n+1} = \frac{a^{n+2}-1}{a-1} \tag{86}$$

$$\frac{a^{n+1}-1}{a-1} + a^{n+1} = \frac{a^{n+2}-1}{a-1} \tag{87}$$

$$a^{n+1} - 1 + a^{n+1}(a-1) = a^{n+2} - 1 \tag{88}$$

$$a^{n+1} - 1 + a^{n+2} - a^{n+1} = a^{n+2} - 1 \tag{89}$$

$$a^{n+2} - 1 = a^{n+2} - 1 \tag{90}$$

Что и требовалось доказать.

1.3.6 #15

Условие. Докажите, что:

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}, n \geqslant 0 \tag{91}$$

Пусть $n = 1$. Тогда:

$$1/2 = 1/2 \tag{92}$$

Теперь пусть формула верна для любых n. Вычислим для $n+1$:

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} + \frac{1}{(n+1)(n+2)} = \frac{n+1}{n+2} \tag{93}$$

$$\frac{n}{n+1} + \frac{1}{(n+1)(n+2)} = \frac{n+1}{n+2} \tag{94}$$

$$\frac{n(n+2)}{(n+1)(n+2)} + \frac{1}{(n+1)(n+2)} = \frac{(n+1)(n+1)}{(n+1)(n+2)} \tag{95}$$

$$\frac{n(n+2)+1}{(n+1)(n+2)} = \frac{n^2+2n+1}{(n+1)(n+2)} \tag{96}$$

$$\frac{n^2+2n+1}{(n+1)(n+2)} = \frac{n^2+2n+1}{(n+1)(n+2)} \tag{97}$$

Что и требовалось доказать.

1.3.7 #16

Условие. Докажите, что $n^3 + 2n$ делится на 3 для $n \geqslant 0$.

Пусть $n = 0$. Тогда выражение равно 0, что условно делится на 3.

Пусть выражение верно для любых n.

Вычислим для $n+1$.

$$(n+1)^3 + 2(n+1) = \tag{98}$$
$$(n+1)(n^2+2n+1) + 2n + 2 = \tag{99}$$
$$n^3 + 2n^2 + n + n^2 + 2n + 1 + 2n + 2 = \tag{100}$$
$$n^3 + 3n^2 + 5n + 3 = \tag{101}$$
$$n^3 + 2n + 3n^2 + 3n + 3 = \tag{102}$$
$$n^3 + 2n + 3(n^2 + n + 1) \tag{103}$$

Представим (103) в виде $A+B$, где $A = n^3+2n$, а $B = 3(n^2+n+1)$. Мы допустили по индукции, что $A \vdots 3$. В то же время B является произведением 3 на натуральное число больше 0, то есть, тоже делится на 3. По свойству делимости сумма чисел, которые делятся на 3, делится на 3.

Что и требовалось доказать.

1.3.8 #17

Условие. Докажите, что дерево с n вершинами имеет в точности $n-1$ рёбер.

Если $n = 0$ то задача не имеет смысла. Если $n = 1$ то дерево с одной вершиной не имеет рёбер, то есть, верно. Если $n = 2$ то между двумя вершинами можно провести только одно ребро, то есть, верно.

Пусть верно для всех n.

Дерево с $n+1$ вершинами это дерево с n вершинами, к которому присоединили ещё одну. Дерево с n вершинами имеет $n-1$ рёбер по индукции.

Так как у нас не граф, а дерево, есть только один способ добавить вершину в дерево: соединив её ровно одним ребром с другой одной существующей вершиной в дереве.

Так как таким образом мы добавили одну вершину и одно ребро, соотношение между количеством рёбер и количеством вершин не изменилось, у нас всё так же ровно на одно ребро меньше, чем вершин.

Что и требовалось доказать.

1.3.9 #18

Условие. Докажите, что сумма кубов первых n положительных целых чисел равна квадрату суммы этих целых чисел, т. е.,

$$\sum_{i=1}^{n} i^3 = \left(\sum_{i=1}^{n} i\right)^2 \tag{104}$$

Ранее в задачах #10 и #12 были доказаны сокращённые формы записи этих сумм. Подставим их в выражение и упростим:

$$\frac{n^2(n+1)^2}{4} = \left(\frac{n(n+1)}{2}\right)^2 \quad (105)$$

$$\frac{n^2(n+1)^2}{4} = \frac{n^2(n+1)^2}{4} \quad (106)$$

Что и требовалось доказать.

1.4 Приблизительные подсчёты

1.4.1 #19

Условие. Содержат ли все ваши книги, по крайней мере, миллион страниц? Каково общее количество всех книг в вашей институтской библиотеке?

Типичное среднее число страниц в книгах в моей библиотеке — 400. Для того, чтобы достичь миллиона страниц, мне нужно иметь минимум $1000000/400 = 2500$ книг. У меня точно нет 2500 книг, поэтому во всех моих книгах точно нет миллиона страниц.

В моей институтской библиотеке было не больше 20 рядов с книгами. Каждый ряд содержал меньше шести полок с каждой стороны ряда. Каждый ряд был длиной меньше пяти метров. Одна книга в среднем 5 см толщиной.

Таким образом, на каждой полке меньше $5/0.05 = 100$ книг. В каждом ряду $100 \times 6 \times 2 = 1200$ книг.

Получается, что в моей институтской библиотеке было не больше $20 \times 1200 = 24000$ книг. Примерно 1000 книг на каждую начальную букву алфавита.

1.4.2 #20

Условие. Сколько слов содержит эта книга?

В русском издании этой книги ровно 720 страниц.

Одна строчка этой книги содержит 7-12 слов, считая формулы, местоимения и части слов, перенесённые с предыдущей строчки, отдельными словами. Возьмём в качестве среднего числа слов на строчку 10.

Число строчек на одной странице примерно 44.

Учитывая то, что в книге активно используются изображения, листинги, выносные формулы и заголовки, уменьшим число строчек на 25%.

Таким образом, в этой книге примерно $720 \times 44 \times 0.75 \times 10 = 7200 \times 33 = 237600$.

Удивительно, но это значит, что в этой книге примерно 240 тысяч слов!

1.4.3 #21

Условие. Сколько часов составляет один миллион секунд? А сколько дней? Выполните все необходимые вычисления в уме.

Как минимум 250 часов.

Это 10 дней и 10 часов, то есть, почти 10 с половиной дней.

1.4.4 #22

Условие. Сколько городов и посёлков в Соединённых Штатах?

Население Штатов $\approx$ 200 миллионов человек. В стране меньше 50 городов с населением больше миллиона человек.

Остаётся 150 миллионов населения.

Если принять размер «города» в 200000 человек населения, а посёлка в 10000, и допустить, что на один город приходится пять посёлков, то получаем следующую систему уравнений:

$$\begin{aligned} 200000x + 10000y &= 150000000 && (107)\\ 5x &= y && (108) \end{aligned}$$

Где x это количество городов, а y это количество посёлков.

Согласно моей модели, получается, что:

$$\begin{aligned} 200000x + 10000 \times 5x &= 150000000 && (109)\\ 20x + 5x &= 15000 && (110)\\ 25x &= 15000 && (111)\\ x &= 600 && (112) \end{aligned}$$

Отсюда $y = 5x = 3000$.

Таким образом, согласно моей упрощённой модели распределения населения по США у них примерно 50 городов-миллионников, 600 городов с населением в среднем 200000 человек, и 3000 посёлков с населением в среднем 10000 человек.

Это много.

1.4.5 #23

Условие. Сколько кубических километров воды изливается из устья Миссисипи каждый день? Не пользуйтесь никакой справочной информацией. Опишите все предположения, сделанные вами для получения ответа.

Миссисипи — широкая река. Широкие реки должны течь медленно.

Скорость истечения воды в m^3/day вычислим так. Перемножим скорость течения воды (на поверхности) на площадь сечения реки.

Допустим, скорость течения широкой огромной реки — как пеший человек, $5km/h$. Допустим, ширина устья Миссисипи — 3 км. Допустим, максимальная глубина 50 м. Допустим, сечение реки это прямоугольник, то есть, площадь сечения это $3km \times 0.05km = 0.15km^2$.

Тогда получается, что скорость истечения равна:

$$\frac{5km \times 0.15km^2}{1 \times h} = \tag{113}$$

$$\frac{0.75km^3}{1/24day} = \tag{114}$$

$$0.75 \times 24\frac{km^3}{day} = \tag{115}$$

$$18\frac{km^3}{day} \tag{116}$$

Итого согласно моим допущениям скорость истечения воды Миссисипи равна 18 кубическим километрам в день.

1.4.6 #24

Условие. В каких единицах измеряется время доступа к жёсткому диску, в миллисекундах или в микросекундах? *в миллисекундах* Сколько времени занимает доступ к слову в оперативной памяти вашего компьютера, больше или меньше микросекунды? *меньше, ближе к нескольким наносекундам* Сколько инструкций может выполнить центральный процессор вашего компьютера в течение года, если компьютер постоянно держать включённым?

Как минимум 1 терафлопс мой Intel Core i7–7700K делает, поэтому получается 1 триллион операций в секунду. $3600 \times 24 \times 365 = 31536000$ секунд в году.

Получается, как минимум 3.1536×10^{19} операций в год.

1.4.7 #25

Условие. Алгоритм сортировки выполняет сортировку 1000 элементов за 1 секунду. Сколько времени займёт сортировка 10000 элементов,

- если время исполнения алгоритма прямо пропорционально n^2?
- если время исполнения алгоритма, по грубым оценкам, пропорционально $n \log n$?

Если сложность алгоритма n^2, то увеличив количество элементов в 10 раз мы увеличили время исполнения в $10^2 = 100$ раз. Получается, что не одну секунду, а 100, почти две минуты.

Если время исполнения алгоритма пропорционально $n \log n$, то значит, новое время исполнения равно:

$$\frac{10^4 \log 10^4}{10^3 \log 10^3} = \tag{117}$$

$$\frac{10 \times 4}{3} = \tag{118}$$

$$13.(3) \tag{119}$$

Таким образом, не 1 секунду, а 14.

1.5 Проекты по реализации

1.5.1 #26

Условие. Реализуйте два эвристических алгоритма решения задачи комивояжёра из раздела 1.1. Какой из них выдаёт на практике более качественные решения? Можете ли вы предложить эвристический алгоритм, работающий лучше любого из них?

Алгоритм 1.

1: **function** NearestNeighbor(P)
2: Из множества P выбираем и посещаем произвольную начальную точку p_0.
3: $p = p_0$
4: $i = 0$
5: **while** Пока остаются непосещённые точки **do**
6: $i = i + 1$
7: Выбираем и посещаем непосещённую точку p_i, ближайшую к точке p_{i-1}
8: Посещаем точку p_i
9: **end while**
10: Возвращаемся в точку p_0 от точки p_{n-1}
11: **end function**

Реализация алгоритма в папке `chapter01`, проект Visual Studio 2017 `01.26.sln`

Алгоритм 2.

1: **function** ClosestPair(P)
2: Пусть n — количество точек множества P.
3: **for** $i \leftarrow n-1, 1$ **do**
4: $d = \infty$
5: **for** Каждая пара точек (s, t), не имеющих общих вершин цепей **do**
6: **if** $dist(s, t) \leqslant d$ **then**
7: $s_m = s$, $t_m = t$ и $d = dist(s, t)$
8: **end if**
9: Соединяем (s_m, t_m) ребром.
10: **end for**
11: **end for**
12: Соединяем две конечные точки ребром.
13: **end function**

Для реализации этих алгоритмов сделаны следующие допущения.

1. Входные данные это массив координат точек на плоскости.
2. Выходные данные это массив координат точек на плоскости в порядке их посещения.

3. Рисовать ничего не будем.

4. Программа будет писать результат в консоль перечислением точек одной под другой.