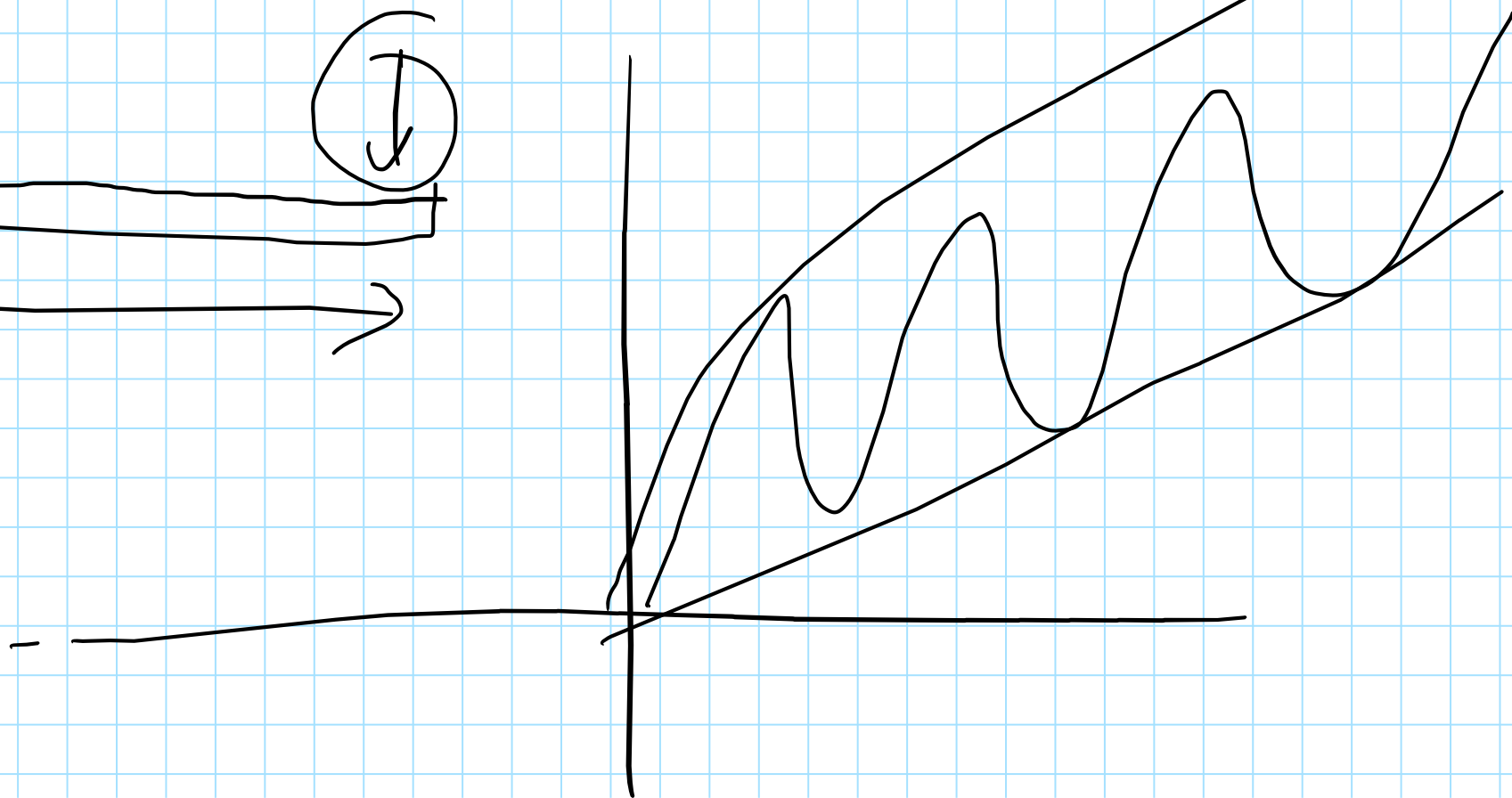
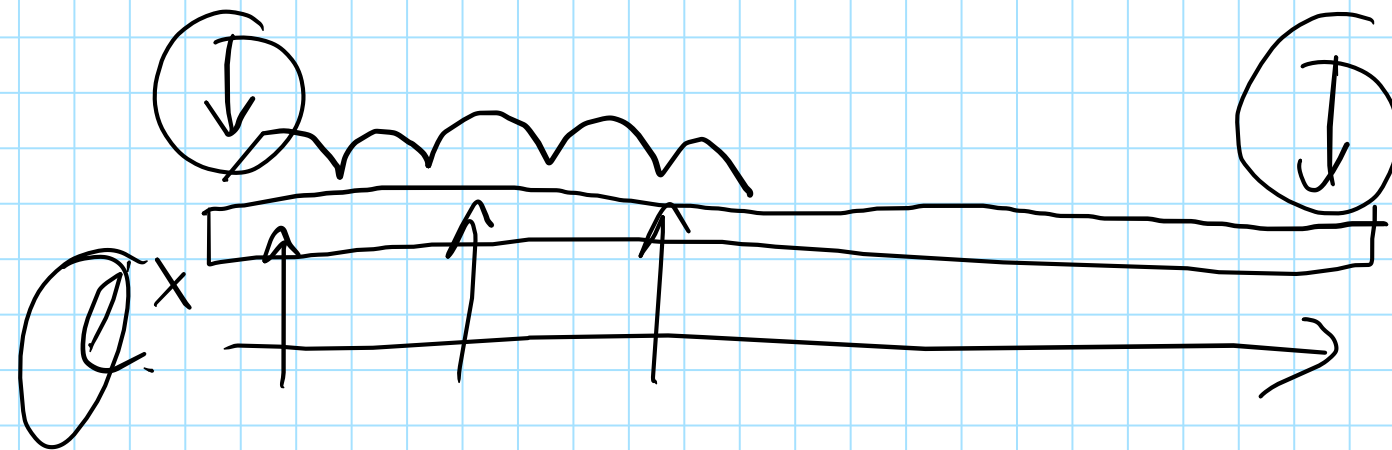


Как нам убедиться в том что найденная наилучшая оценка сверху(или снизу) действительно наилучшая? Как доказать что лучше оценки нет?

$$O(n^k)$$

$$O(\cdot) = \Omega(\cdot) \Rightarrow \Theta(\cdot)$$

Объясните пожалуйста, почему определение длины массива это $O(1)$ по времени. Если в языках со строгой типизацией типа c++ при инициализации массива заранее задается размер и тип элементов, то стандартный список в питоне это динамический массив, который может содержать ссылки на объекты разных типов. Вычисление длины такого массива это тоже $O(1)$ по времени?



$$\text{len}(S) = n$$

6. Палиндромом называется строка, которая одинаково читается слева направо и справа налево. Рассмотрим некоторую строку S состоящую из маленьких латинских букв. Какое минимальное количество букв (возможно нуль) нужно в ней удалить, чтобы она не являлась палиндромом (если это возможно)? Предложите алгоритм решения задачи, докажите его корректность, оцените сложность.

- 1) $abba$ — пал.
 2) abb — не пал.
 3) a — пал.
 $\underbrace{aaa \dots a}_n$
 4) $\underbrace{a \dots a}_n \underbrace{b \dots b}_n \underbrace{a \dots a}_n$

ответ

1

0

∞

1

$$\begin{array}{c} \rightarrow \\ \leftarrow n \end{array} \Rightarrow O(n)$$

1) не пал \rightarrow ответ "0"

2) пал $\forall a \in \Sigma \Rightarrow \underbrace{a \dots a}_n \quad \forall a \rightarrow \text{ответ } \infty$

3) пал \forall кот есть хотя бы 2 раз. Буквы

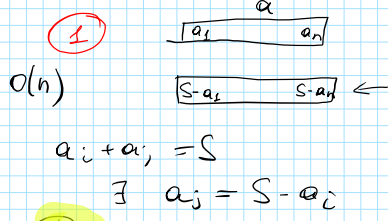
\downarrow
ответ "1"

пусть a, b .

$\underbrace{a \dots a}_i \underbrace{b}_{i+1} ? \quad \underbrace{b}_{i+1} \underbrace{a \dots a}_i$

i -й слева: b
 i -й справа: $a \Rightarrow$ не пал.

3. На вход задачи поступают массив $a = [a_1, \dots, a_n]$ и число S . Требуется проверить, есть ли в массиве элементы a_i и a_j ($i \neq j$), такие что $a_i + a_j = S$.
1. Постройте алгоритм, решающий задачу за $O(n \log n)$. **Бреинг**
2. Представьте, что у вас есть структура данных «множество», добавление элементов в которую стоит $O(1)$ и проверка наличия элемента в структуре так же стоит $O(1)$. Решите задачу, используя эту структуру за $O(n)$.
3. Постройте алгоритм, решающий задачу за линейное время, при условии, что массив a отсортирован, используя дополнительно $O(1)$ памяти.



1 2 5 0 -1

For i 1.. n

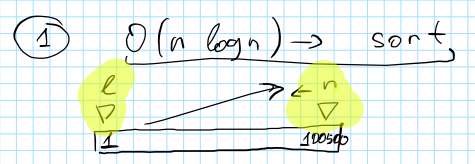
for j i+1.. n

$O(n^2)$ - Трив.

$S = 4 \rightarrow \text{Yes}$

$S = 8 \rightarrow \text{No}$

$n + n + n + \dots = n^2$



$O(n)$

while $l < r$:

$a[l] + a[r] > S$:

$r = r - 1$

$a[l] + a[r] < S$:

$l = l + 1$

$a[l] + a[r] == S$:

return Yes

return No

3. На вход задачи поступают массив $a = [a_1, \dots, a_n]$ и число S . Требуется проверить, есть ли в массиве элементы a_i и a_j ($i \neq j$), такие что $a_i + a_j = S$.
1. Постройте алгоритм, решающий задачу за $O(n \log n)$.
2. Представьте, что у вас есть структура данных «множество», добавление элементов в которую стоит $O(1)$ и проверка наличия элемента в структуре так же стоит $O(1)$. Решите задачу, используя эту структуру за $O(n)$.
3. Постройте алгоритм, решающий задачу за линейное время, при условии, что массив a отсортирован, используя дополнительно $O(1)$ памяти.

~~$B = \{a_1, \dots, a_n\}$~~

~~for $i = 1..n$:~~

~~if $S - a_i \in B$:~~

~~return Yes~~

~~return No~~

$O(n)$

$a_i + a_j = S$

$\exists a_j = S - a_i$

$a_i = \frac{S - a_i}{2}$

$a = [\dots, \frac{S - a_i}{2}]$

$S = 2$

$a = [1, 2, 3]$

$S - a_i = 1$

$B = \{y\}$

for $i = 1..n$:

if $S - a_i \in B$:

return Yes

$B.add(a_i)$

return No

$B = \{a_i, \dots, y\}$

\parallel

$S - a_i$

5. Дан массив a из n положительных целых чисел. Нужно разбить этот массив на максимальное количество непрерывных подмассивов так, чтобы после сортировки элементов внутри каждого подмассива весь массив стал отсортированным.

Более формально, нужно выбрать индексы i_1, \dots, i_{k-1} ($i_k = n$), такие что массив разбивается на подмассивы

$$[a_1, \dots, a_{i_1}], [a_{i_1+1}, \dots, a_{i_2}], \dots, [a_{i_{k-1}+1}, \dots, a_{i_k}],$$

в результате сортировки каждого из которых (и обратной склейки) получается отсортированный исходный массив; количество блоков разбиения k должно быть максимально возможным.

Предложите $O(n \log n)$ алгоритм для решения этой задачи, можно использовать $O(n)$ дополнительной памяти.

Обязательно обоснуйте работоспособность вашего алгоритма. Оцените его сложность и объём необходимой дополнительной памяти. Мы будем признательны, если описание алгоритма будет дано не листингом кода, а просто текстом, возможно, с привлечением псевдокода.

а: 2 1 4 3 → 1 2 | 3 4 → 1 2 3 4 (2)
 1 2 3 4 → 1 | 2 | 3 | 4 → 1 2 3 4 (4)
 4 3 2 1 → 1 4 3 2 1 → 1 2 3 4
 а 2 1 4 3
 s 1 2 3 4

① $s = \text{sort}(a)$ $O(n \log n)$

② а: 2 1 4 3

s: 1 2 3 4

а: 2 1 4 3

s: 1 2 3 4

i совм. max. → n-об

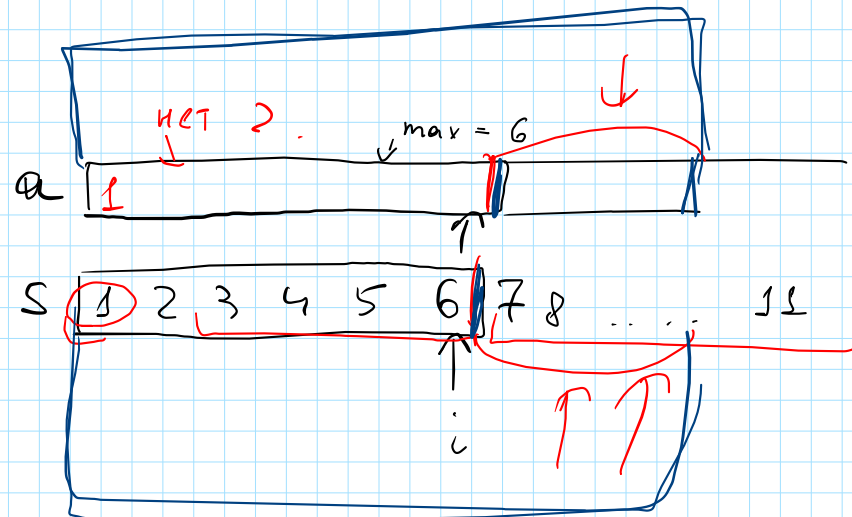
$$\text{set}(a[1..i]) = \text{set}(s[1..i]) \rightarrow \text{per no } i$$

$$d[2] = 1$$

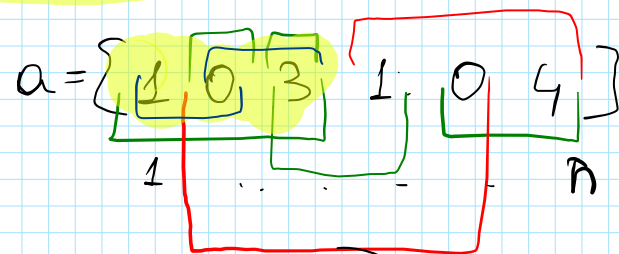
$$d[1] = -1$$

$$\max(a[1..i]) = \max(s[1..i])$$

$$d[2] = (\text{кон-но } 2 \text{ в } a) - (\text{кон-но } 2 \text{ в } s)$$



2. Дан массив $[a_1, \dots, a_n]$ неотрицательных целых чисел и число K . Отрезок $[a_l, \dots, a_r]$ назовём интересным, если в нём не более одного нулевого элемента, а сумма всех элементов не превосходит числа K . Найдите число интересных подотрезков массива.



$0, n$

$$K = 4$$

Трив:

$0, n$

$$O(n^3)$$

$$O(n^2)$$

$$O(n^2)$$

$$\#0 = 1$$

$$C_n^2 \quad \text{Sum} = 4$$

$$O(n^2)$$

$$\frac{n(n-1)}{2} + n$$

$[l, r]$

$$l \geq 2$$

$$l = 1$$

пог-к $[i, i]$ - инт, то.
 $\forall [l, r]$ - инт.
 $i \leq l \leq r \leq j$

$$l = 1$$

$$S = 0$$

$$C = 0$$

for $r = 1 \dots n$

$$S += a[r]$$

$$C += (a[r] == 0)$$

while $C > K$ or $C > 1$:

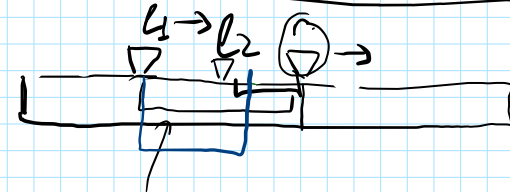
$$S -= a[l]$$

$$C -= (a[l] == 0)$$

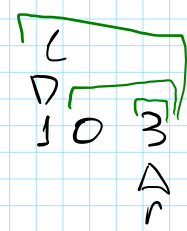
$$l += 1$$

$$ans += (r - l + 1)$$

↑



инт



$$K = 4$$

$$S = 4$$

$$C = 1$$

$$ans = 1 + 2 + 3$$