Асимптотики

О-нотация

- Способ оценить время работы алгоритма без привязки к железу и деталям реализации
- Различают временную и пространственную сложность

Попытка строго сформулировать

- Можно задать функцию $f(n)$: время работы при размере входных данных n
- Может быть очень сложной, например: $f(n) = 3n^2 + 4\log(2n+7) + 18$
- Такая точность не всегда нужна, хочется обобщить

- Введем *достаточно простую* $g(n)$, назовем ее асимптотикой изначальной функции, это значит, что она ограничивает ее сверху
- Чтобы это понять, возьмем некоторое $C > 0$, и некоторый размер входных данных n_0
- Тогда $\forall n > n_0$ верно $f(n) \leq C \cdot g(n)$
- Для примера выше подойдет $C = 100, g(n) = n^2$
- Тогда говорят $f(n) = O(n^2)$

Попытка наглядно показать

- ```
  x = 5
  y = x * 2
  ```
  2 операции, $O(1)$
- ```
  for i in range(n):
    print(i)
  ```
  n операций, $O(n)$

- ```
  s = 0
  for i in range(100 * n):
  ```
  ```

```
  s += i
print(s)
```
$100n + 2$ операции, $O(n)$

- ```
  for i in range(n):
    for j in range(n):
        print(i * j)
        print(i + j)
    print(i)
  ```
  $2n^2 + n$ операций, $O(n^2)$

- ```
  a = [i for i in range(n)]
  a.reverse()
  ```
  Зависит от устройства reverse, $O(n)$.

- Для определения асимптотики времени/памяти какого-то алгоритма обычно берут максимально простые функции, если это возможно (по определению асимптотики)
- Есть стандартные асимптотики алгоритмов и стандартные названия этих асимптотик: единица/константа, линия, $n \log n$, квадрат, куб. Такие вещи надо помнить, потому что это облегчает общение.

Квадратичные сортировки

- Асимптотика $O(n^2)$
- Редко применяются на практике
- Важно знать, чтобы понимать принципы упорядочивания

Сортировка выбором

- Идея: мы можем n раз выбрать минимум и составить отсортированный массив
- Поиск минимума выполняется в зависимости от размера массива:
  - На первом шаге минимум надо выбрать из n элементов и поставить его на позицию 1
  - На втором шаге из $n - 1$ и поставить его на позицию 2
  - ...
  - На n-м шаге мы его сразу знаем

- [5,6,2,3] $\to$ [2,6,5,3] $\to$ [2,3,5,6] $\to$ [2,3,5,6] $\to$ [2,3,5,6]
- $n + (n-1) + \ldots + 1 = \frac{n(n+1)}{2} = O(n^2)$

Сортировка вставками

- Идея: двигаем элемент влево, пока он не встанет на свое место
- Представим, что у нас слева есть отсортированный участок, пусть его длина k:
  - Возьмем $k+1$-й элемент
  - Будем двигать его, пока он не встанет на своё место
  - Получили отсортированный участок размера $k+1$
- [5,6,2,3] $\to$ [5,6,2,3] $\to$ [5,6,2,3] $\to$ [2,5,6,3] $\to$ [2,3,5,6]
- $1 + 2 + \ldots + n = \frac{n(n+1)}{2} = O(n^2)$

Сортировка пузырьком

- Каждый раз будем смещать самый "тяжелый" элемент в конец
  - Для всех i от 1 до $n-1$, если $a_i > a_{i+1}$ обмениваем их местами
  - Для одного элемента в худшем случае n обменов
  - Для n элементов n^2
  - Асимптотика $O(n^2)$
- Можно понять, что после i шага последние i элементов отсортированы
- Пример **одного прохода**: [5,3,6,2] $\to$ [3,5,6,2] $\to$ [3,5,6,2] $\to$ [3,5,2,6]

Линейные структуры данных

Вектор

- Динамический массив
- размер ($size$) и вместимость ($capacity$)
- Пока $size \leq capacity$, просто добавляем
- Затем выделяем новый массив с $capacity' = C \cdot capacity$, копируем туда содержимое старого
- Пример $C = 2$:

    - [1] (size 1, capacity 1)
    - [1,2] (size 2, capacity 2, перекладываем)
    - [1,2,3] (size 3, capacity 4, перекладываем)
    - [1,2,3,4] (size 4, capacity 4)
    - [1,2,3] (size 3, capacity 4)
- Пусть, итоговая $capacity$ вектора равна n. Тогда сумма capacity всех массивов не превосходит $n + \frac{n}{2} + \frac{n}{4} + \ldots + 1 \leq 2n = O(n)$.
- n операций сделали за $O(n)$
- На практике используют разные C, там тоже получается сумма геометрической прогрессии, которая зависит только от C.

Стек

- Добавить элемент на стек
- Узнать "верхний" элемент стека
- Удалить "верхний" элемент
- Можно реализовать используя обычный массив
- Пример использования: проверить, является ли данная последовательность скобок "правильной"
    - (())(()())
    -))()
    - Если скобка открывается, добавляем ее в стек
    - Если скобка закрывается и стек не пуст, удаляем
    - В конце проверим, что стек пуст
- n операций сделали за $O(n)$

Очередь

- Добавить элемент в конец очереди
- Удалить элемент из начала очереди
- Узнавать первый/последний элементы
- Сделать n таких операций за $O(n)$
- Можно реализовать, используя массив и указатель на начало очереди
    - лишняя память

Очередь

- Можно реализовать, используя 2 стека, представить как два стакана, которые касаются донышками
  - При добавлении элемента кладем его в правый стек
  - При удалении удаляем из левого
    - Если в левом пусто, перекладываем туда по одному все элементы из правого. Они развернутся.
    - Каждый элемент переложится только 1 раз $\Rightarrow$ асимптотика $O(n)$
- Пример:][$\rightarrow$][A $\rightarrow$][AB $\rightarrow$][ABC $\rightarrow$ ABC][$\rightarrow$ BC][$\rightarrow$ BC][D