Сортировки

Параллель С
26.09.2020

Организационные моменты

- Лекции и контесты. Возможно, когда-нибудь будут офлайн занятия.
- Ревью кода
- Перед новым годом - коллоквиум
- Весной - зачет
- Дисттуры
- Теоретические задания - листочки
- Дедлайн на контест - коллоквиум/зачет. Число сданных задач влияет на итог

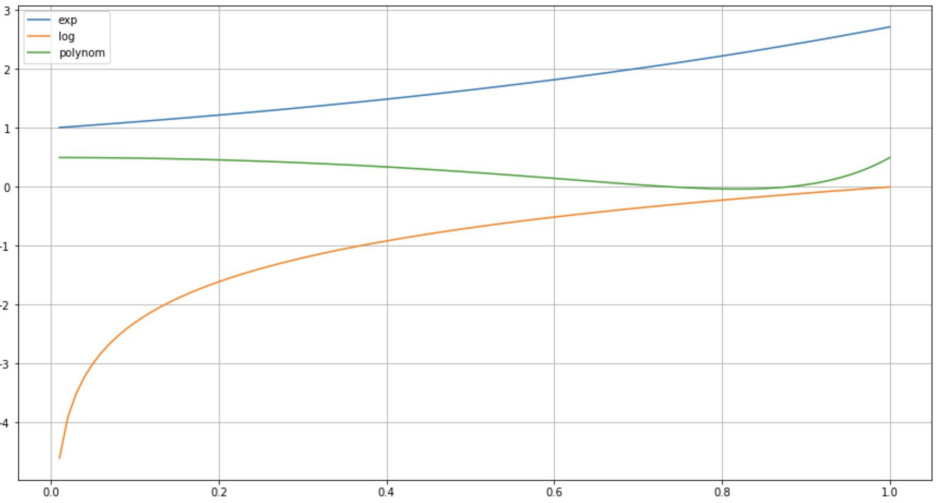
Оценка сложности алгоритмов

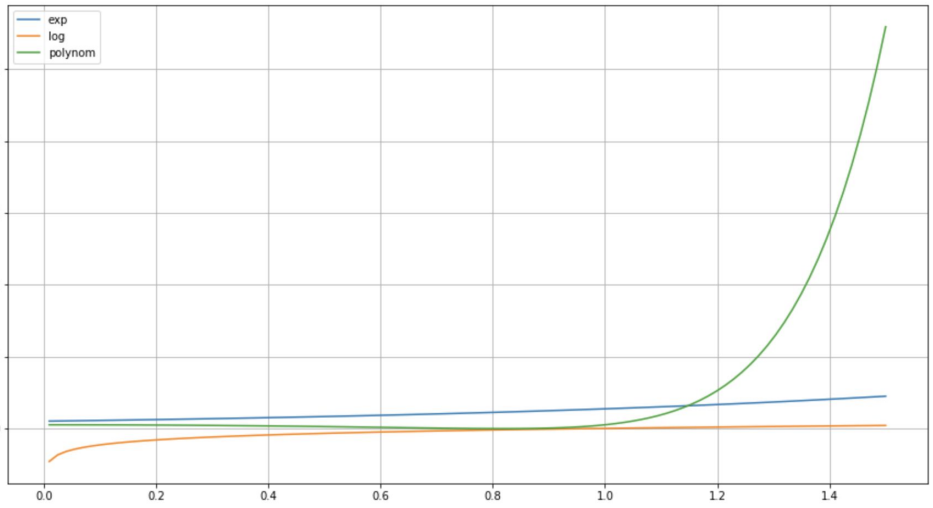
- Надо уметь оценивать время, за которое отработает алгоритм. Нам важно примерное число операций, а не точное.
- Пример: найти минимальный элемент в массиве
  - Отсортировать массив по возрастанию, взять 1-й элемент
  - Пройтись по массиву и обновлять значение текущего минимума

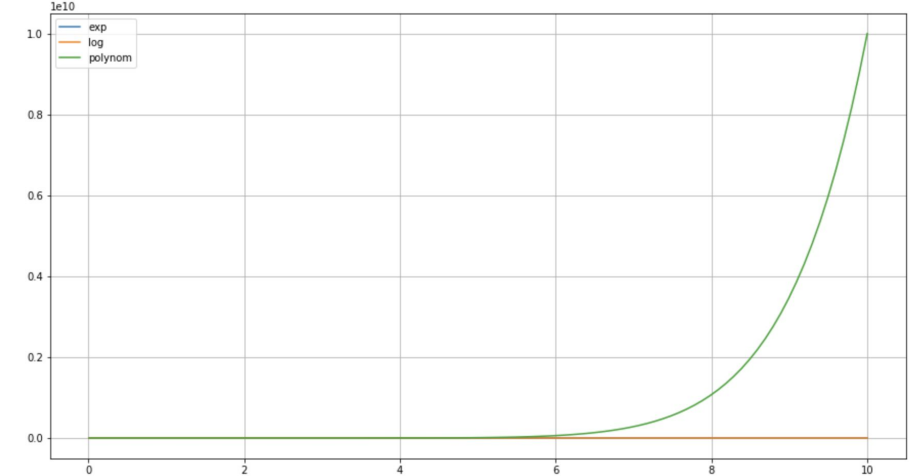
```
cur_min = a[0];
for i=1...n
    cur_min = min(cur_min, a[i]);
return cur_min;
```

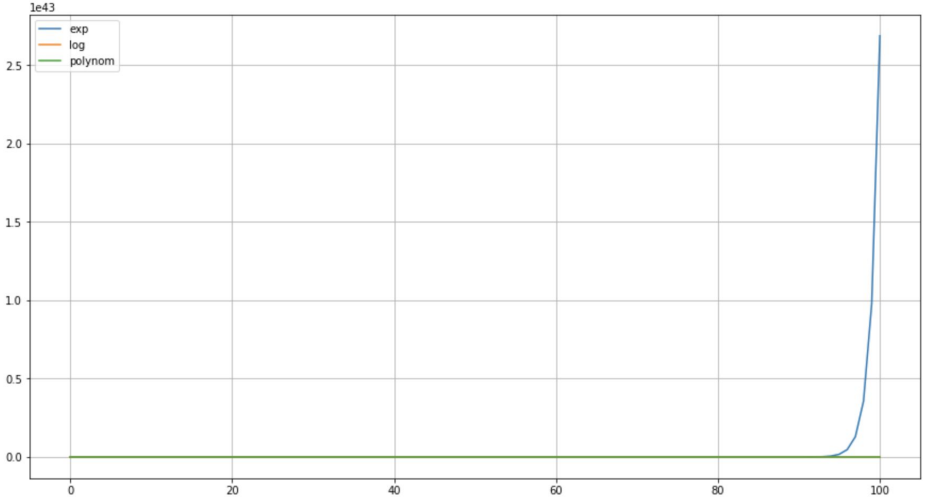
```
sort(a);
return a[0];
```

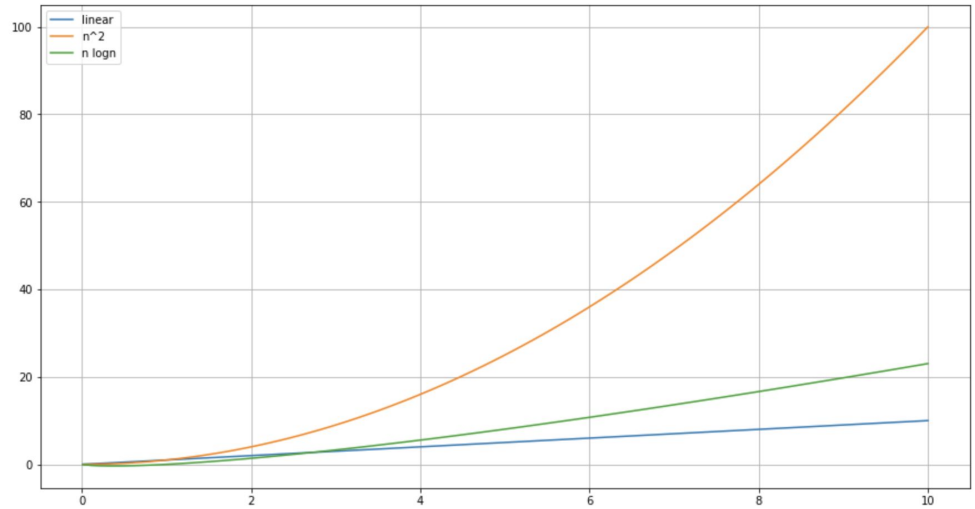
- *n vs n^2*


exp
log
polynom


exp
log
polynom


1e10
exp
log
polynom

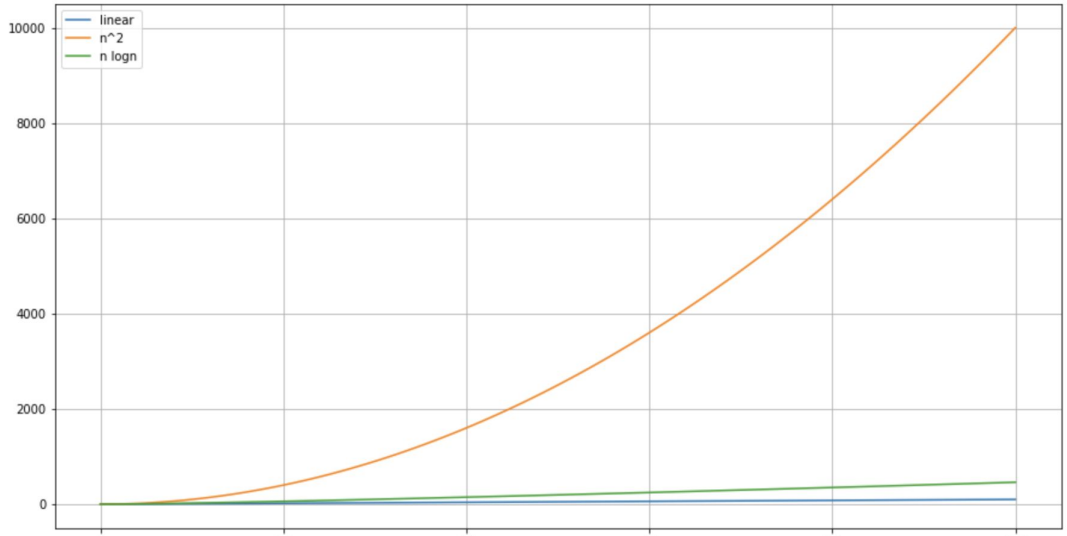

1e43
exp
log
polynom



O(n)

O(n^2)

O(n log n)



Более формально

f = O(g), если найдется такое N и константа C, что при любых n > N

f(n) <= Cg(n)

Примеры:

3n + 2n^2 = O(n^2)

e^x + xlog x = O(e^x)

x + y = O(x + y)

123456782020 = O(1)

100n = O(n)

Два вида сортировок

Нужен только оператор сравнения

- Пузырек
- Вставками
- Выбором
- Сортировка слиянием
- qsort

Нужны какие-то специфичные знания про объекты

- Сортировка подсчетом
- Поразрядная сортировка (radix sort)

Что значит *a* < *b*

a, b - в целом почти любые объекты, для которых можно придумать сравнение

Требования к сравнению:

- Из a < b, b < c следует a < c
- Для любых a, b либо a < b, либо b < a, либо a = b (то есть не должно быть таких объектов a, b, что одновременно a < b и b < a)

Сортировка вставками

- Пусть на i-й итерации у нас есть отсортированный префикс длины i
- Вставим следующий элемент на нужную позицию в этот префикс за O(N)
- Переходим к следующей итерации

Сортировка выбором

- Пусть на i-й итерации первые i минимальных элементов стоят на своих местах
- Выберем минимум среди оставшихся элементов и поставим на i-ю позицию
- Перейдем к следующей итерации

Сортировка пузырьком

- Делаем N проходов
- Проходимся по всем элементам массива слева направо и, если текущий больше следующего, меняем их местами
- После каждого прохода как минимум один элемент встанет на свое место (и больше с него не сдвинется)

Сортировка подсчетом

- Хотим отсортировать массив a
- Заведем массив cnt, где cnt[i] будет отвечать за количество элементов в a, равных i
- Теперь достаточно пройтись по cnt по возрастанию i и выписать в массив элемент i cnt[i] раз