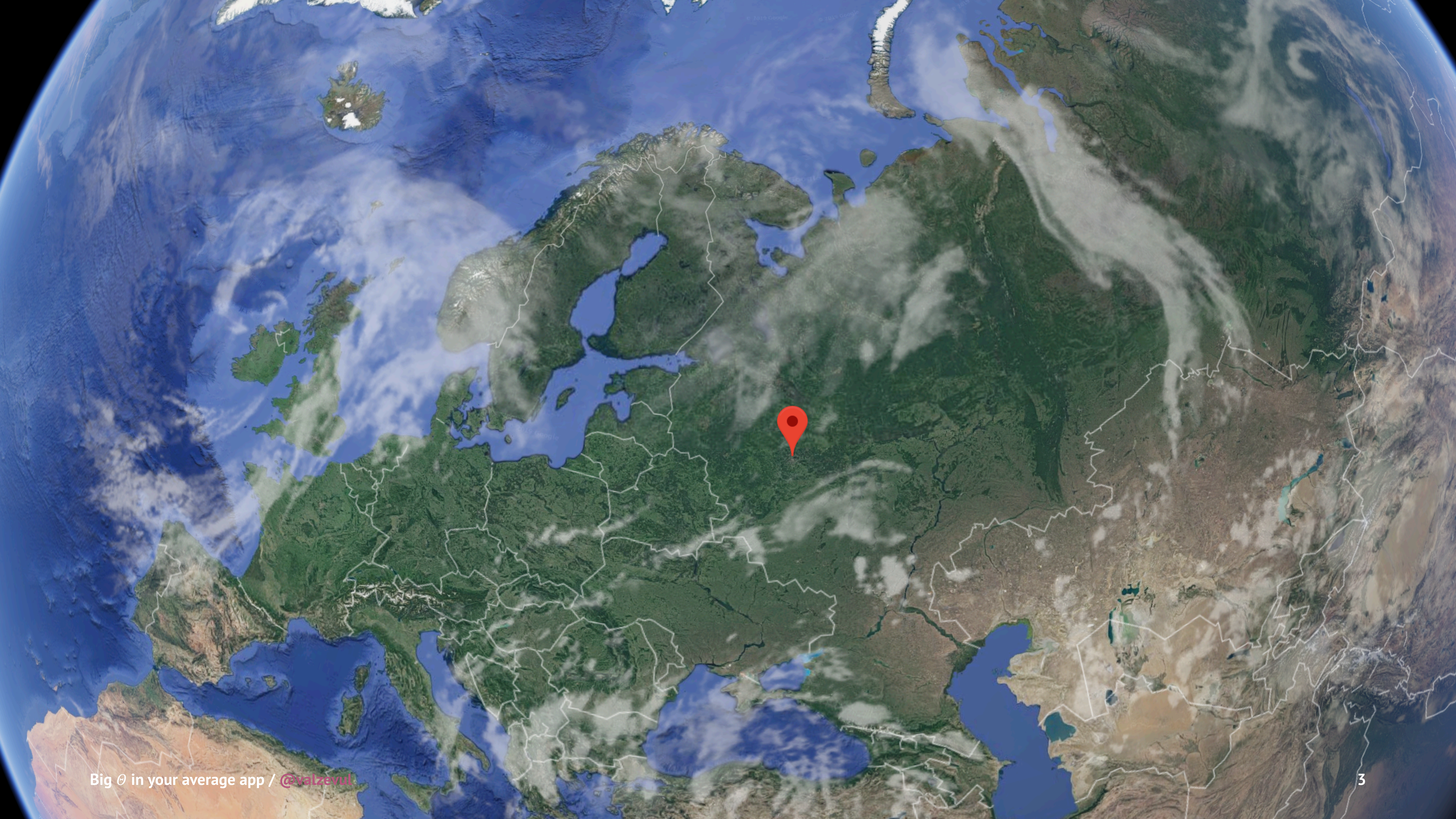
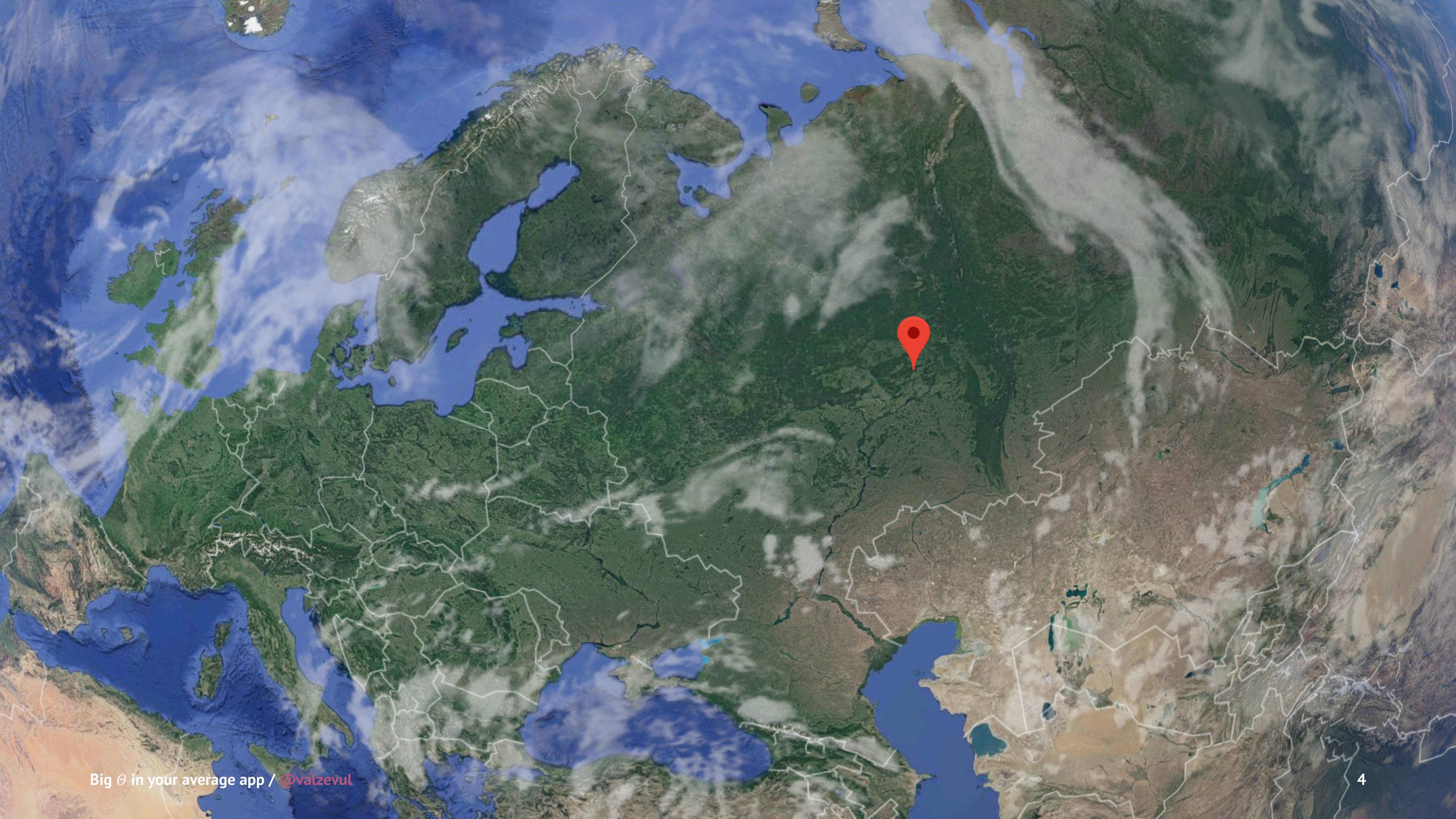
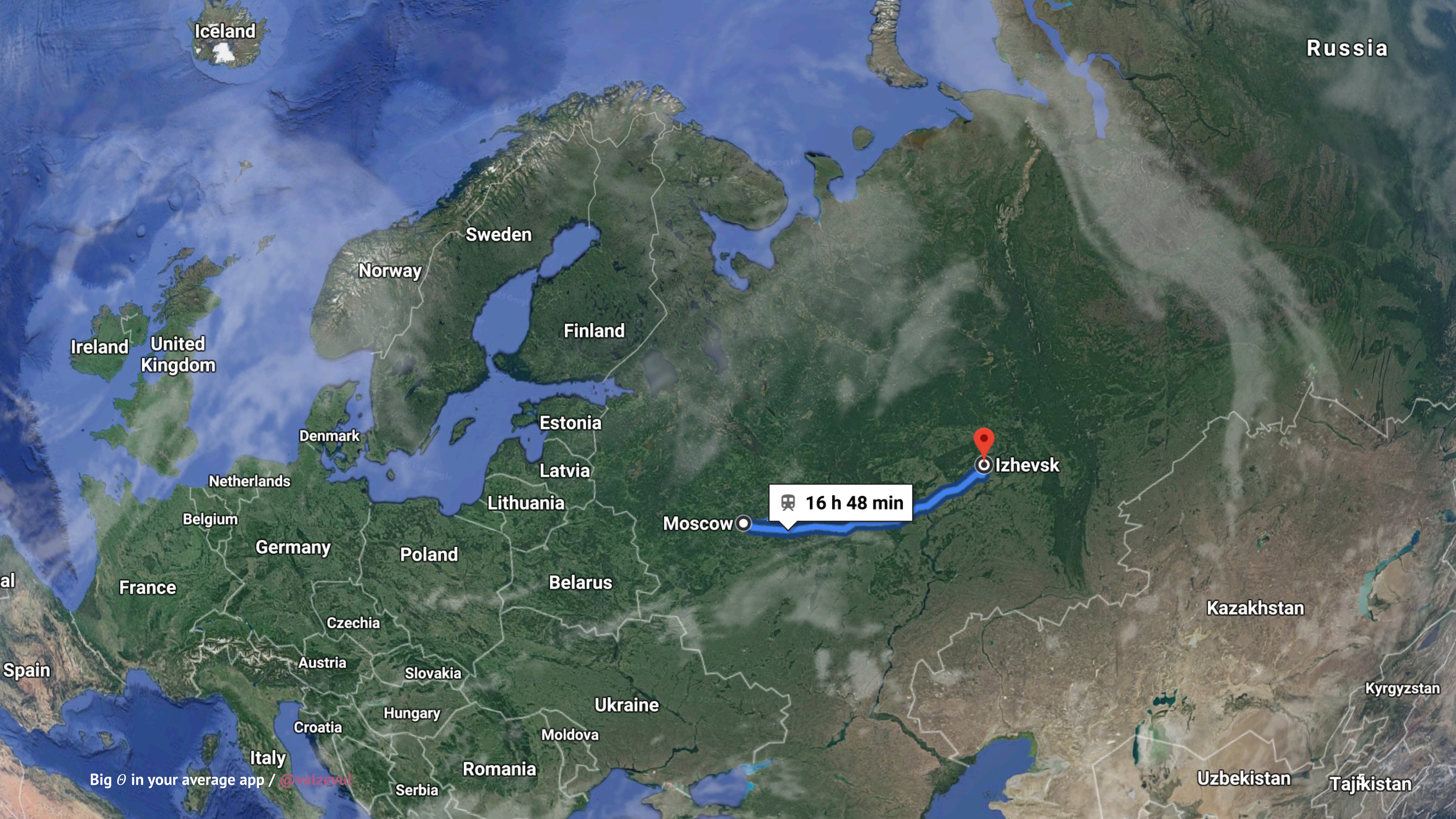
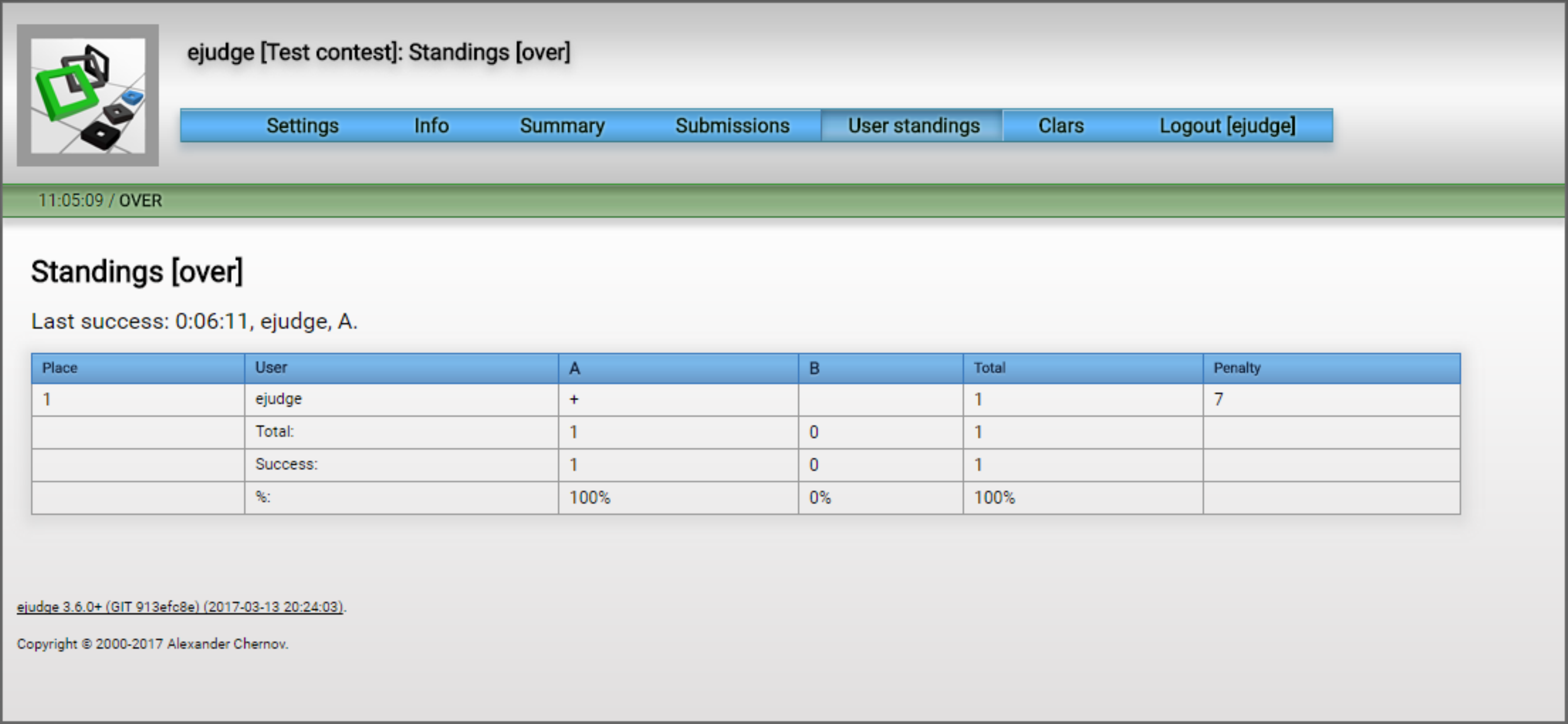
Big θ Notation in your Average App

Vadim Drobinin | @valzevul






Iceland
Russia
Sweden
Norway
Finland
Ireland
United Kingdom
Estonia
Denmark
Izhevsk
Latvia
Netherlands
Lithuania
16 h 48 min
Moscow
Belgium
Germany
Poland
al
France
Belarus
Kazakhstan
Czechia
Spain
Austria
Slovakia
Kyrgyzstan
Ukraine
Hungary
Croatia
Moldova
Italy
Romania
Uzbekistan
Tajikistan
Serbia



ejudge [Test contest]: Standings [over]

11:05:09 / OVER

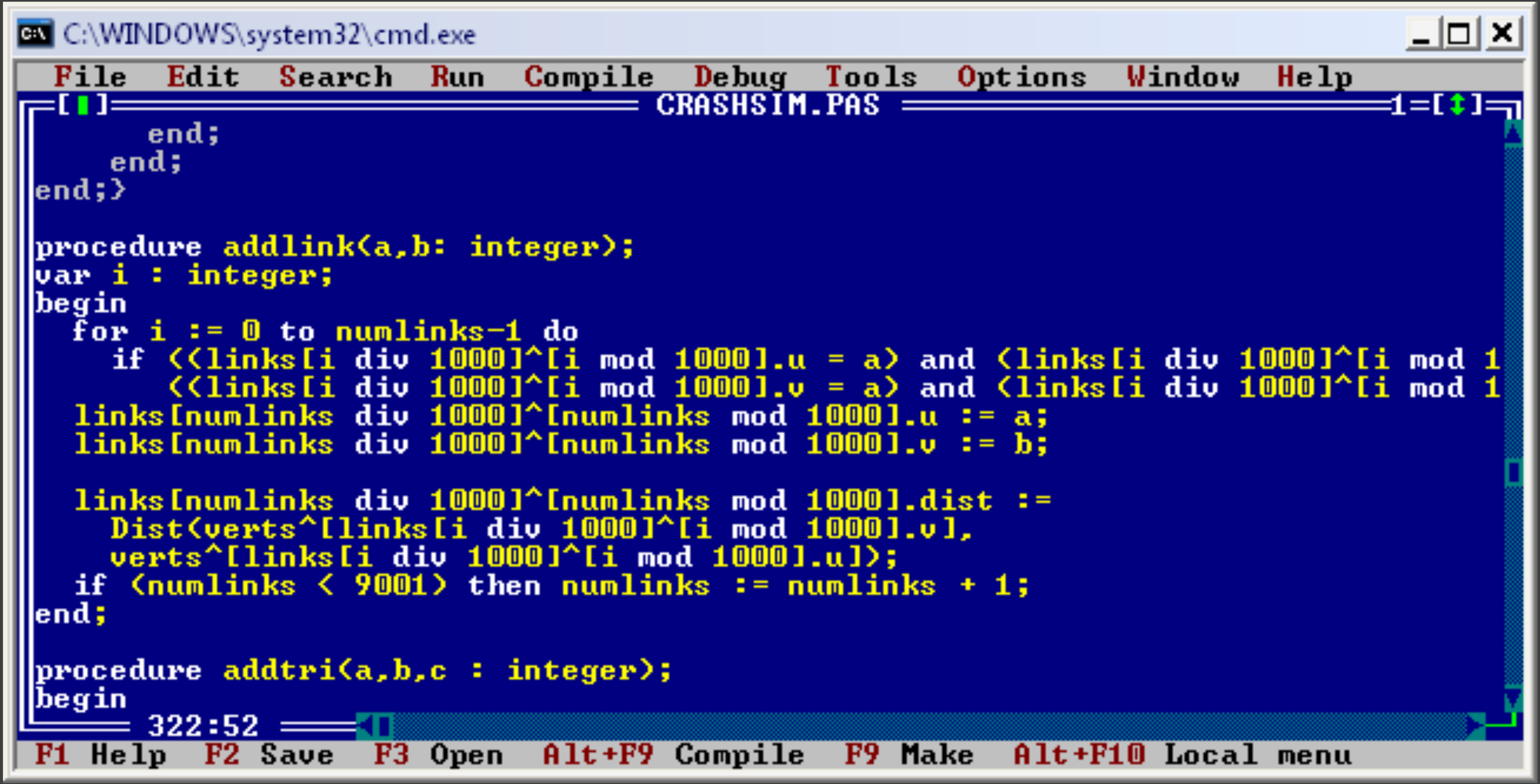
Standings [over]

Last success: 0:06:11, ejudge, A.

Place	User	A	B	Total	Penalty
1	ejudge	+		1	7
	Total:	1	0	1	
	Success:	1	0	1	
	%:	100%	0%	100%	

ejudge 3.6.0+ (GIT 913efc8e) (2017-03-13 20:24:03).

Copyright © 2000-2017 Alexander Chernov.


C:\WINDOWS\system32\cmd.exe
File Edit Search Run Compile Debug Tools Options Window Help
CRASHSIM.PAS
1
322:52
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu

```
      end;
    end;
end;}

procedure addlink(a,b: integer);
var i : integer;
begin
  for i := 0 to numlinks-1 do
    if ((links[i div 1000]^[i mod 1000].u = a) and (links[i div 1000]^[i mod 1
       ((links[i div 1000]^[i mod 1000].v = a) and (links[i div 1000]^[i mod 1
  links[numlinks div 1000]^[numlinks mod 1000].u := a;
  links[numlinks div 1000]^[numlinks mod 1000].v := b;

  links[numlinks div 1000]^[numlinks mod 1000].dist :=
    Dist(verts^[links[i div 1000]^[i mod 1000].v],
    verts^[links[i div 1000]^[i mod 1000].u]);
  if (numlinks < 9001) then numlinks := numlinks + 1;
end;

procedure addtri(a,b,c : integer);
begin
```

tl;dr

- What's the Big O? Or is it Big θ?
- Do you ***really*** need algorithms?
- If you do though, which ones?
- Common myths (or sad truth?)
- Big O for your app

- What's Big O Notation?

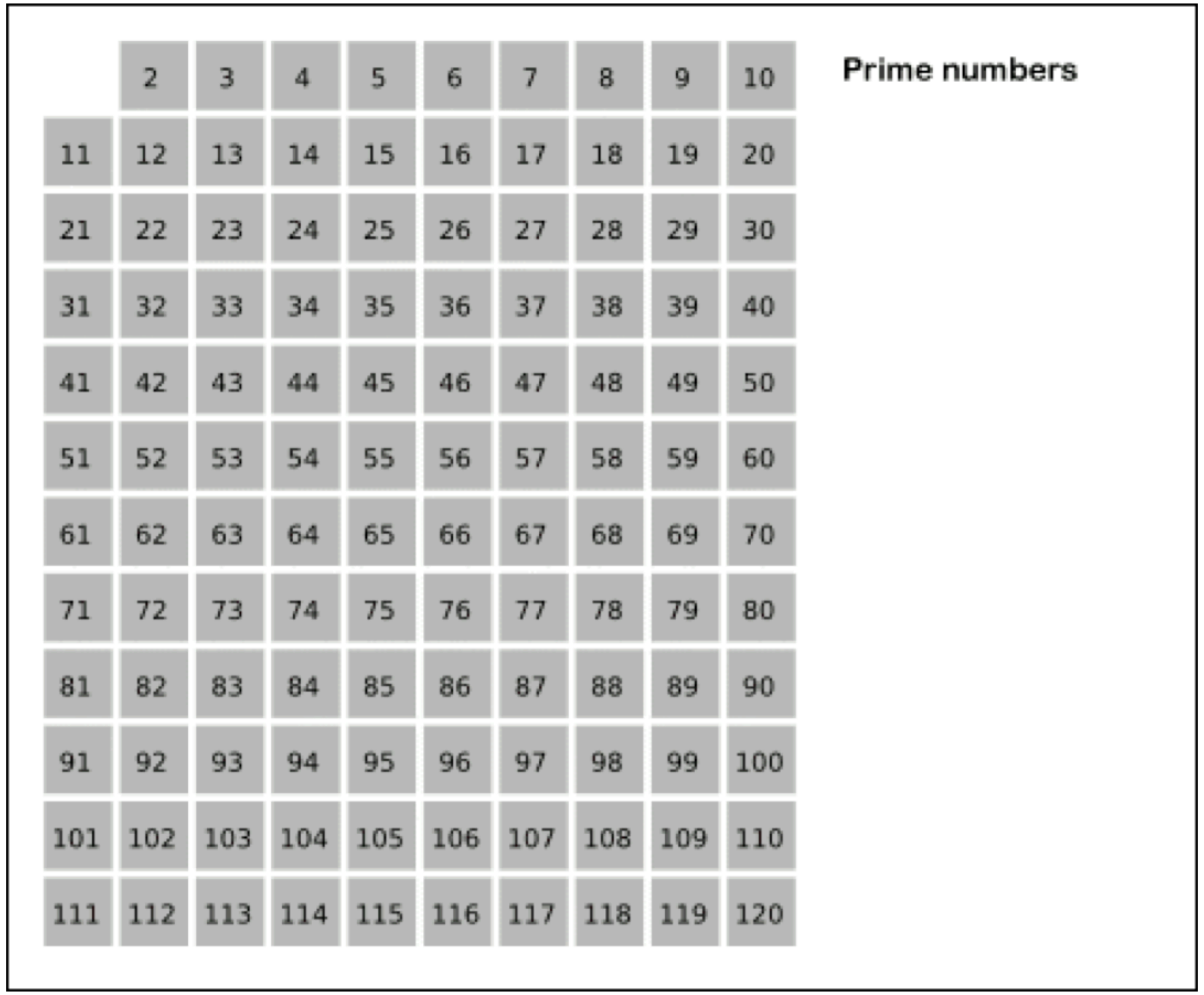
— A way of describing the efficiency.

Big θ ≤ Big O

- Big O is an upper bound.
- Big θ is a tight bound (upper ***and*** lower).

Big θ ⩽ Big O

- O(The Sieve of Eratosthenes)?
- ✅ O(nlogn)
- ❌ θ(nlogn)
- ✅ θ(nloglogn)

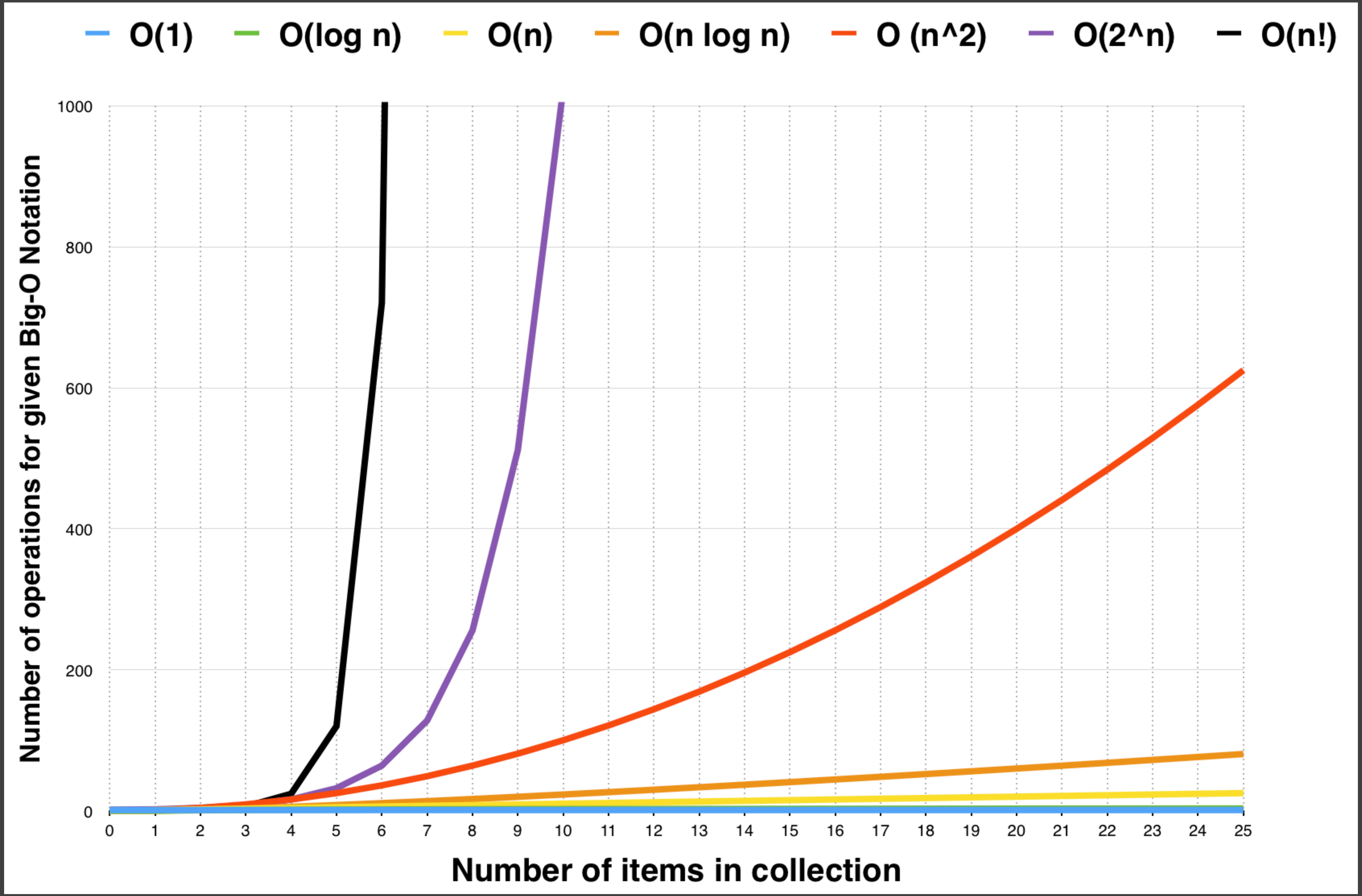


Prime numbers

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Examples

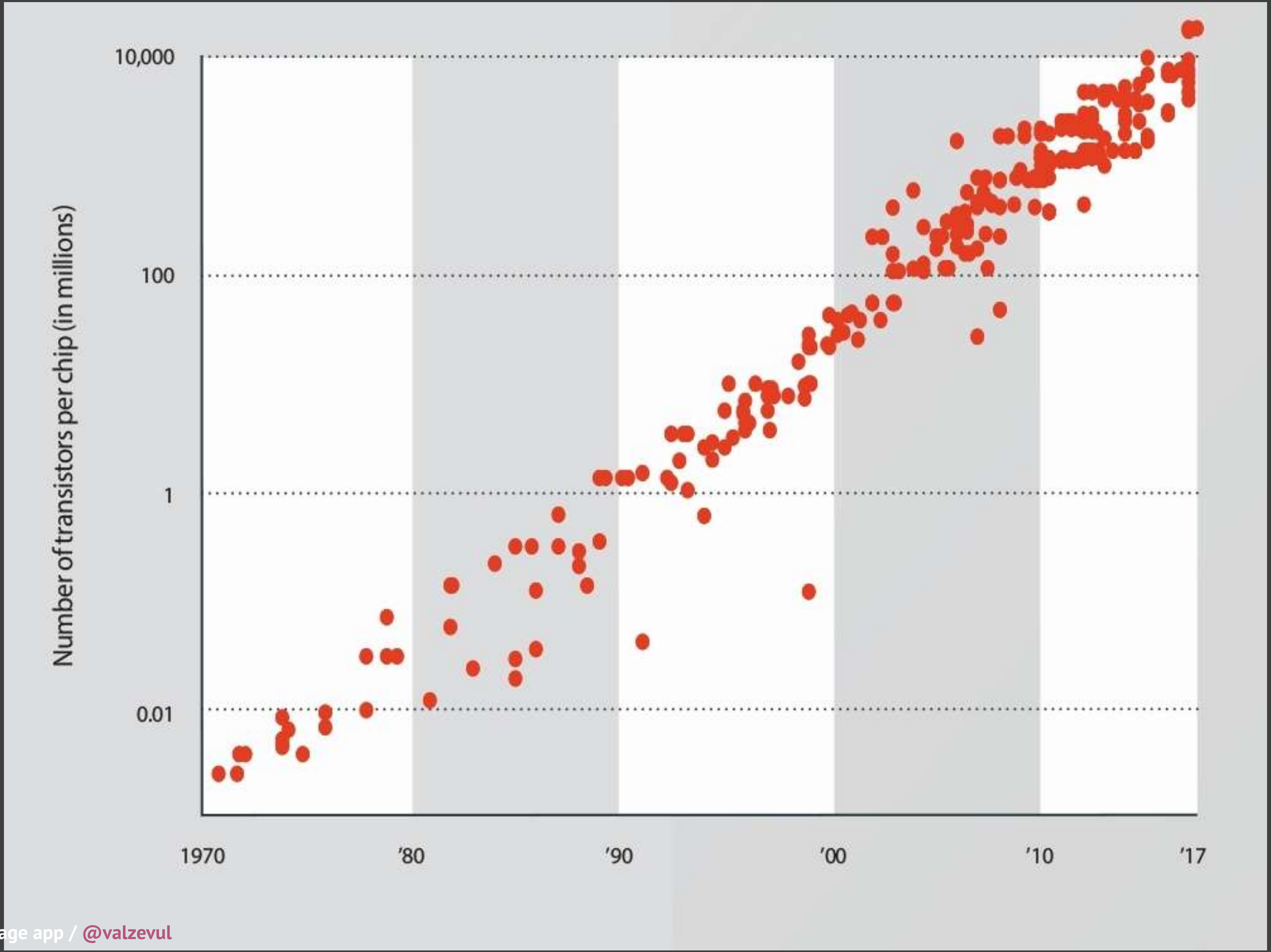
Big-O	Description
O(1)	the best
O(log n)	pretty great
O(n)	good performance
O(nlogn)	decent performance
O(n^2)	kinda slow
O(n^3)	poor performance
O(2^n)	very poor performance
O(n!)	intolerably slow



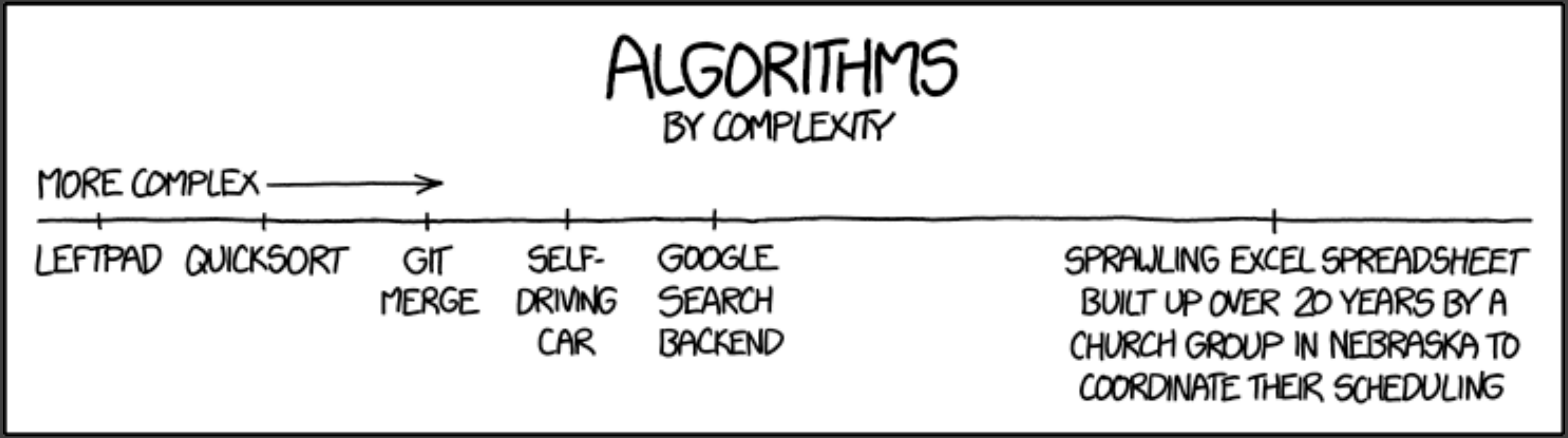
© raywenderlich.com

"Modern devices are way too powerful for users to notice a difference between ***kinda slow*** and ***decent performance*** algorithms".

— Some developers


Number of transistors per chip (in millions)
10,000
100
1
0.01
1970
'80
'90
'00
'10
'17

> Should everyone know how to implement \*(algorithm name)* if they develop a mobile app?
>
> — CS-grads


ALGORITHMS
BY COMPLEXITY
MORE COMPLEX
LEFTPAD
QUICKSORT
GIT MERGE
SELF-DRIVING CAR
GOOGLE SEARCH BACKEND
SPRAWLING EXCEL SPREADSHEET BUILT UP OVER 20 YEARS BY A CHURCH GROUP IN NEBRASKA TO COORDINATE THEIR SCHEDULING

© xkcd

Examples

- Store data from API
- Path-finding algorithm to draw connections
- Snap an object to the nearest edge
- Geohashes to optimize local storage
- Hashing / Cryptography
- Low-level performance issues (ie dropping frames)

> "All really useful algorithms are already implemented in default libraries. If they are not, you can find them on Github or StackOverflow."
>
> — Some developers

Binary Search from StackOverflow

```swift
public func binarySearch<T: Comparable>(_ a: [T], key: T) -> Int? {
    var lowerBound = 0
    var upperBound = a.count
    while lowerBound < upperBound {
        let midIndex = (lowerBound + upperBound) / 2
        if a[midIndex] == key {
            return midIndex
        } else if a[midIndex] < key {
            lowerBound = midIndex + 1
        } else {
            upperBound = midIndex
        }
    }
    return nil
}
```

Binary Search from StackOverflow

```swift
public func binarySearch<T: Comparable>(_ a: [T], key: T) -> Int? {
    var lowerBound = 0
    var upperBound = a.count
    while lowerBound < upperBound {
        let midIndex = lowerBound + (upperBound - lowerBound) / 2
        if a[midIndex] == key {
            return midIndex
        } else if a[midIndex] < key {
            lowerBound = midIndex + 1
        } else {
            upperBound = midIndex
        }
    }
    return nil
}
```

> Algorithms are essential to whiteboard interviews.
>
> — Terrified students

> Is there an “O” of your app and can we define it?
>
> — Curious developers

Your app's very own Big O

- Write down experience and assign Big Os
- Decide whether they were adding up to each other or the most terrible issue overcome minor once
- Use it to measure the memory consumption as well
- Multiple Os if the former, and add them up if the latter
- O(n^2) + O(n) = O(n^2 + n) = O(n^2) - kinda slow
- O(n^2) * O(n) = O(n^3) - poor performance

Summary

- Remember the difference between Big O and Big Θ
- Be careful copying code from the Internet
- Optimizing algorithms is not the only way to care about your users
- Learn algorithms because they're fun
- Try to evaluate your app UX and performance using Big O notation as a reference

Questions?

drobinin.com | @valzevul