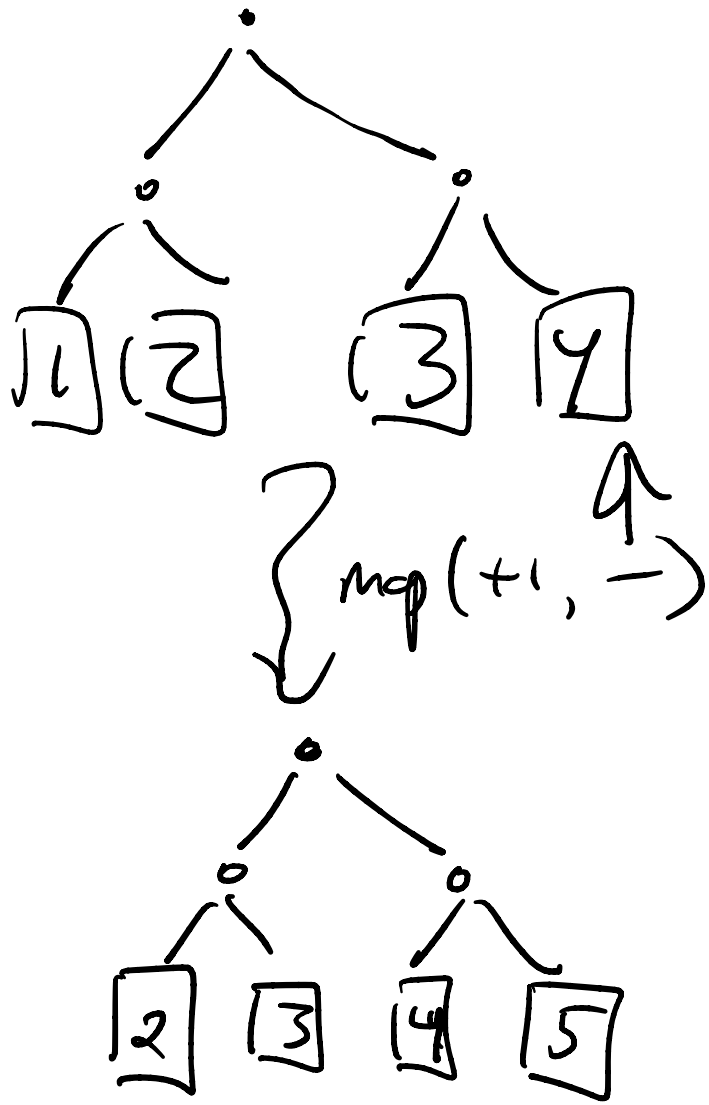
Lecture 21: Data into your program

1. getting data from files
2. map + reduce =

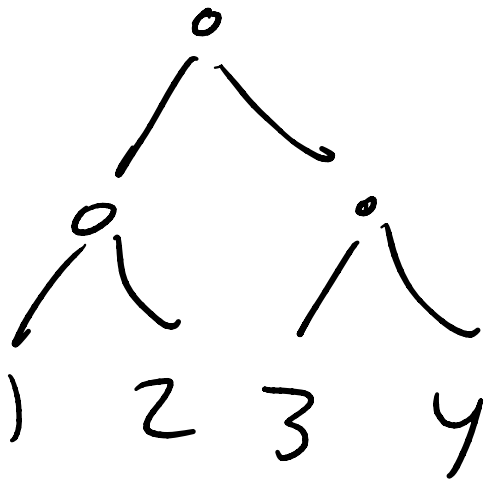
   map reduce
   - ↳ Files
   - ↳ Networked data
3. extract-combine (?)

Combining map and reduce

```
fun map(f, t) =
    case t of
        Empty => Empty
      | Leaf x => Leaf(fx)
      | Node(l, r) =>
        Node(map(f, l),
             map(f, r))
```



```
fun reduce (n, e, t) =
  case t of
    Empty => e
  | Leaf x => x
  | Node (l, r) =>
      n( reduce(n, e, l),
         reduce(n, e, r))
```



reduce(+, 0)

1 + 2 + 3 + 4

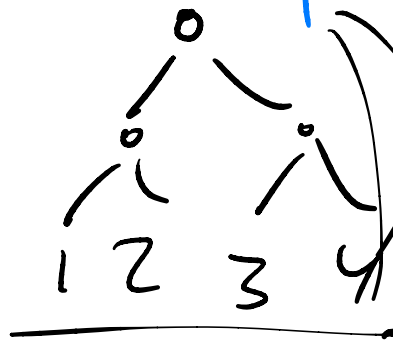
= 10

Deforestation =

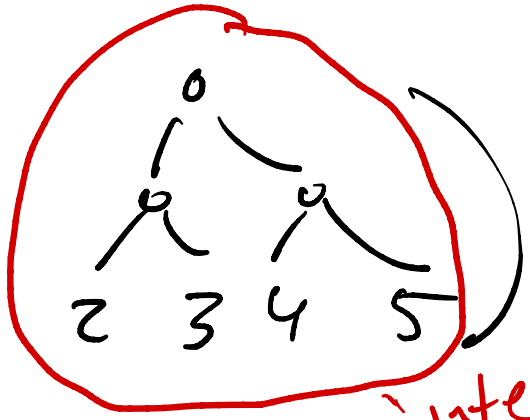
eliminating intermediate trees

O(n) work O(n)

reduce (+, 0, map (fn x => x+1,



↦ reduce (+, 0



"intermediate tree"

↦ (2+3)+(4+5)

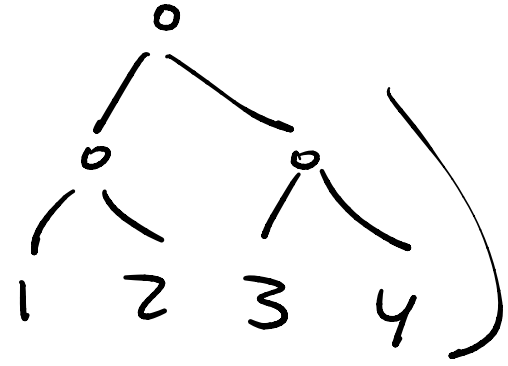
uses O(n) space

= 14

(* Idea/spec mapreduce(f, e, n, t) =
reduce(n, e, map(f, t)) *)

```
fun mapreduce(f, e, n, t) =
    case t of
        Empty => e
      | Leaf x => f(x)
      | Node(l, r) => n(mapreduce(f,e,n,l),
                        mapreduce(f,e,n,r))
```

map reduce (fn x => x+1, 0, +,



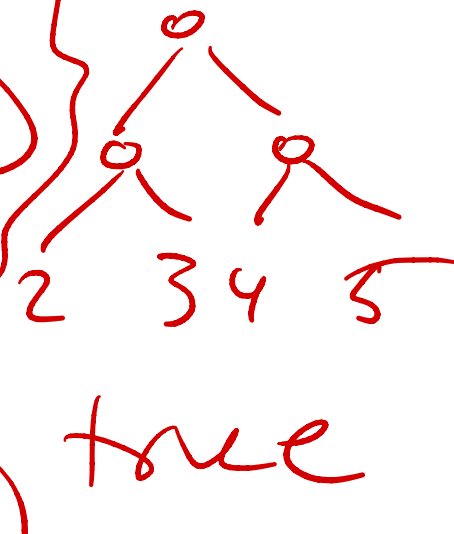
⟼ (1+1) + (2+1) + (3+1) + (4+1)

⟼ 2 + 3 + 4 + 5

⟼ 14

don't make true

O(n)



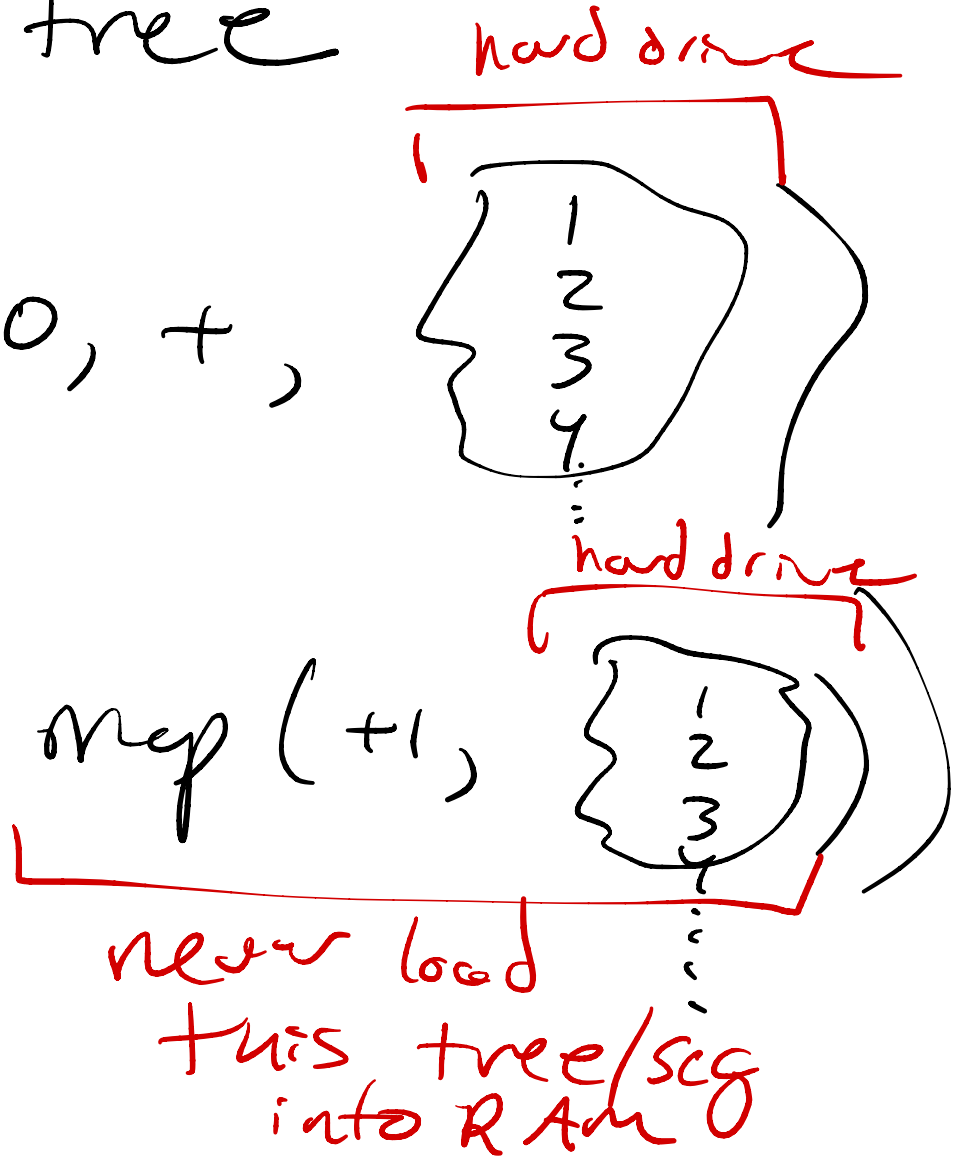
true

Imagine that data is in a file rather than a tree

mapreduce(+1, 0, +,

vs

reduce(+, 0, map(+1,



Signature MAP_REDUCE =

sig

```
  type 'a mapreducable

  val mapreduce :
        ('a -> 'b)
      * 'b
      * ('b * 'b -> 'b)
      * 'a mapreducable
      -> 'b
```

end

f

e

n

"tree"

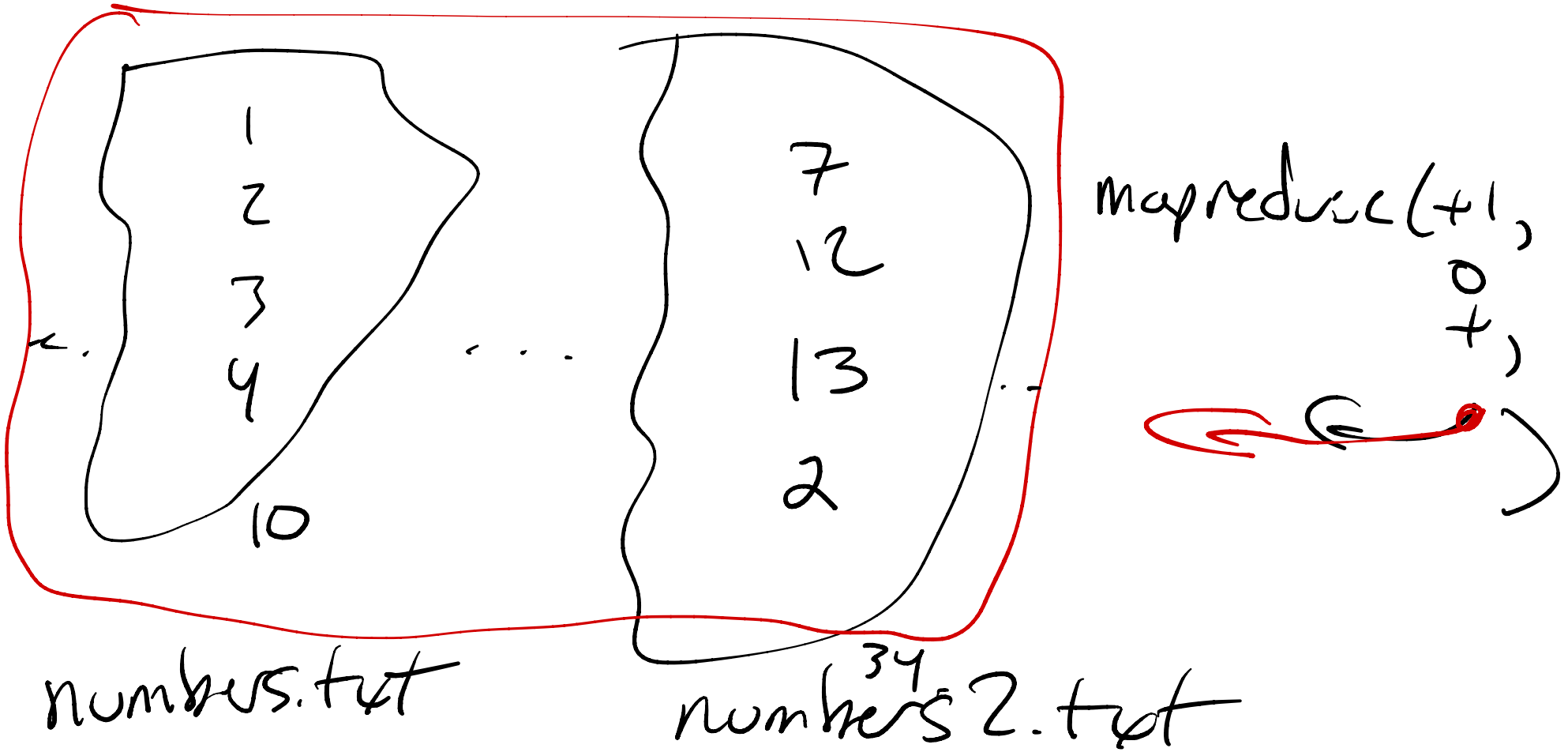
Examples

not deforested

1. Sequences with mapreduce = reduce after map
2. files with a parsing function

Lab 10

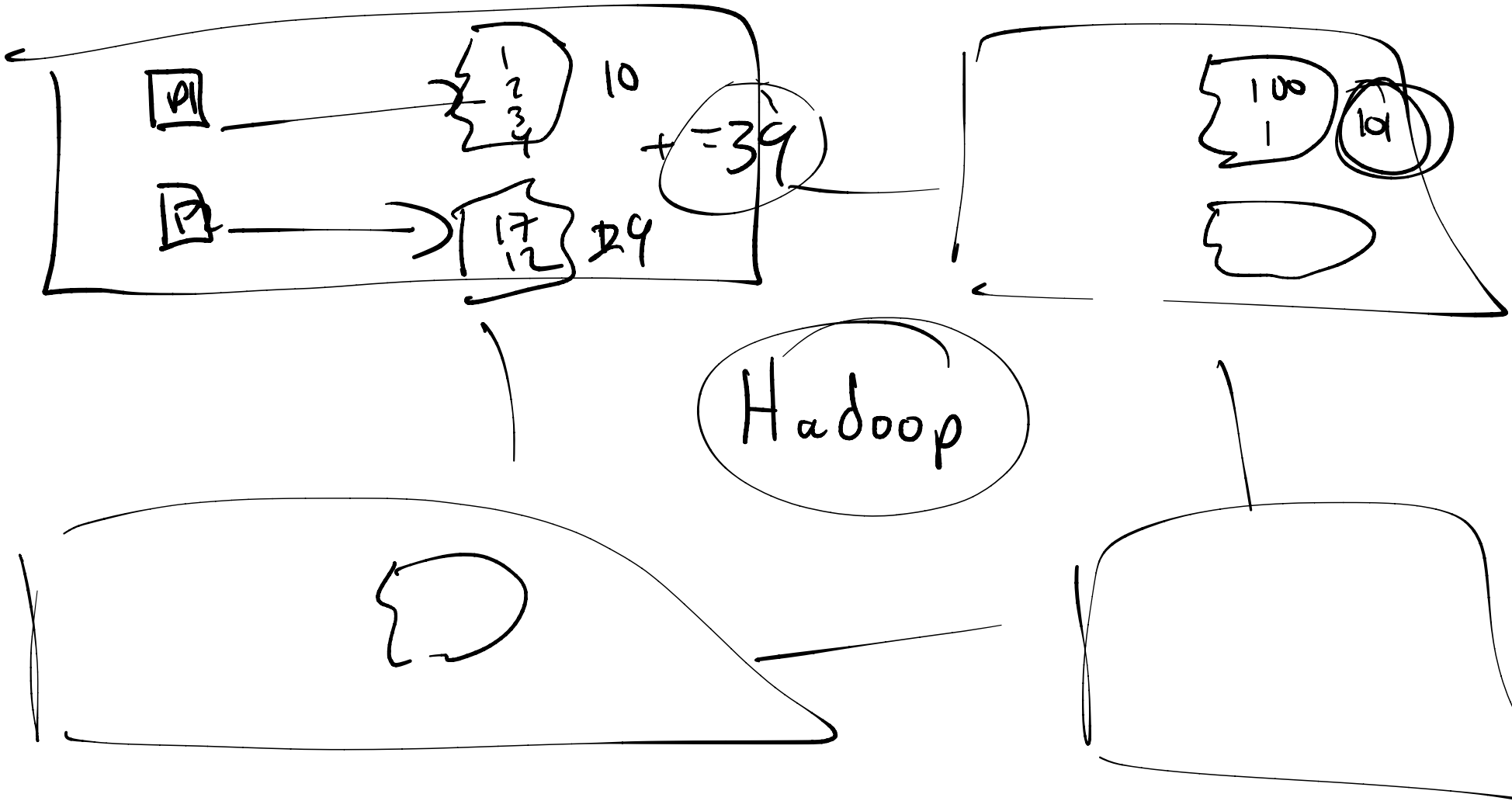
3. many files
4. many files on many computers



Idea: ① compute the summary (sum) of each file

② add summaries for each file

= 44



Idea: ① each computer does the summary for its files

② add/summarize all the computers

frequency counts via

map/reduce

naive Bayes