Design and Analysis of Algorithms

CS202

Today’s class

- Asymptotic Notation
- Local minimum exercises
- Loop Invariants

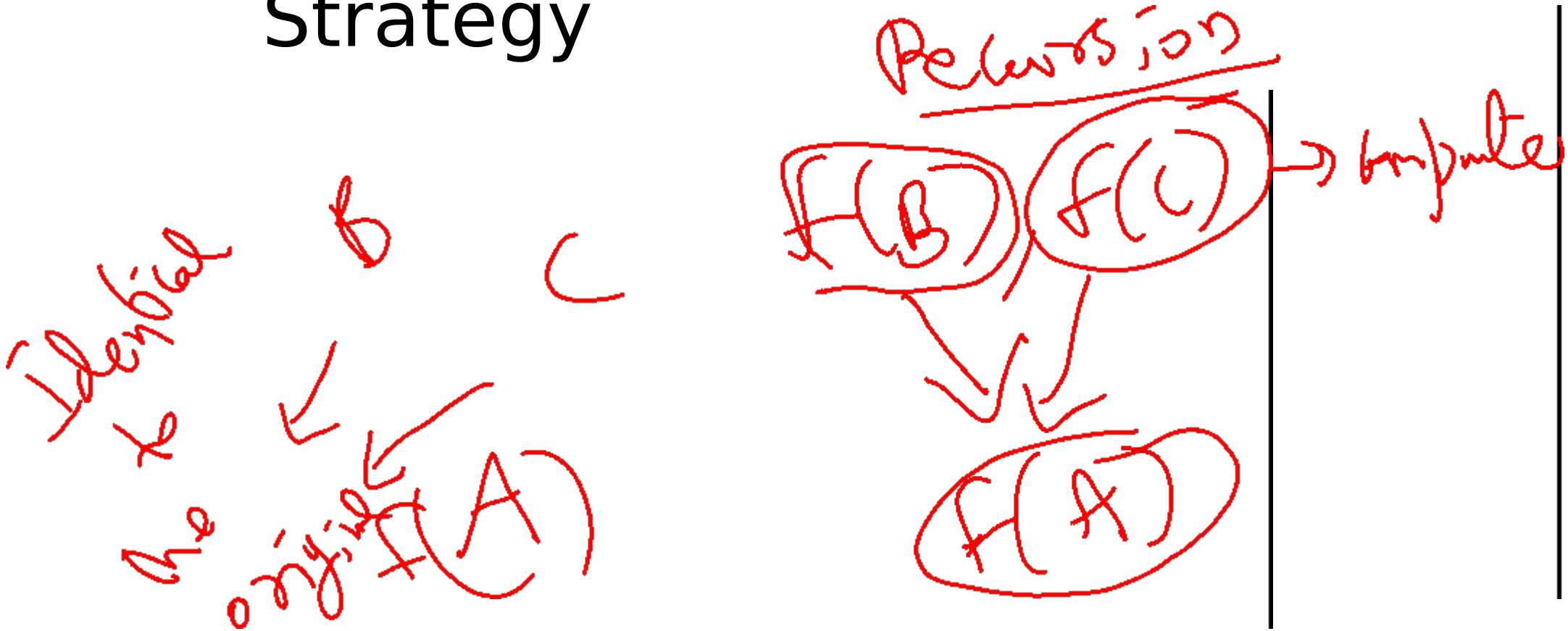
Find the input size

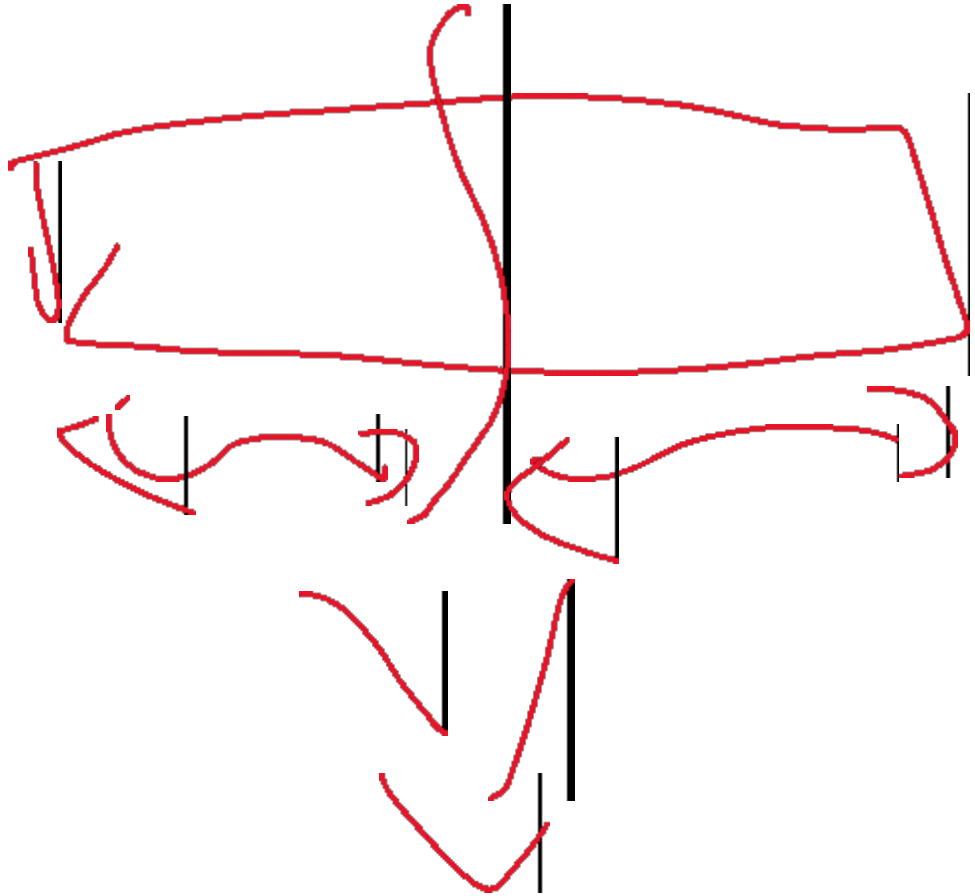
1. Input: x, y in {1,2,...,n}. Output: xy
2 log_2 n

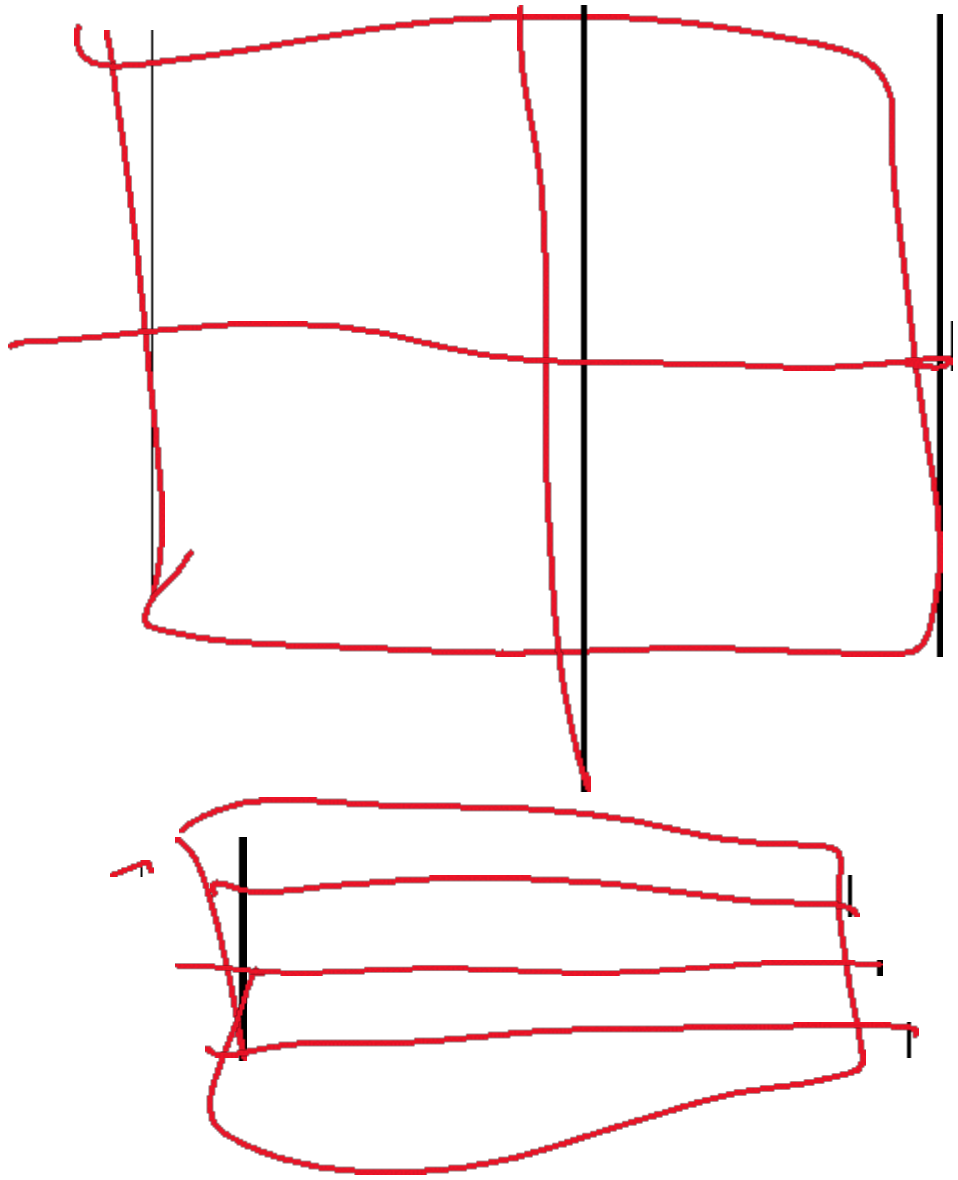
2. Input: A,B: n by n matrix. Output: AB
(n^2)

Input size: 2n^2 (number of elements)*

Divide & Conquer Strategy



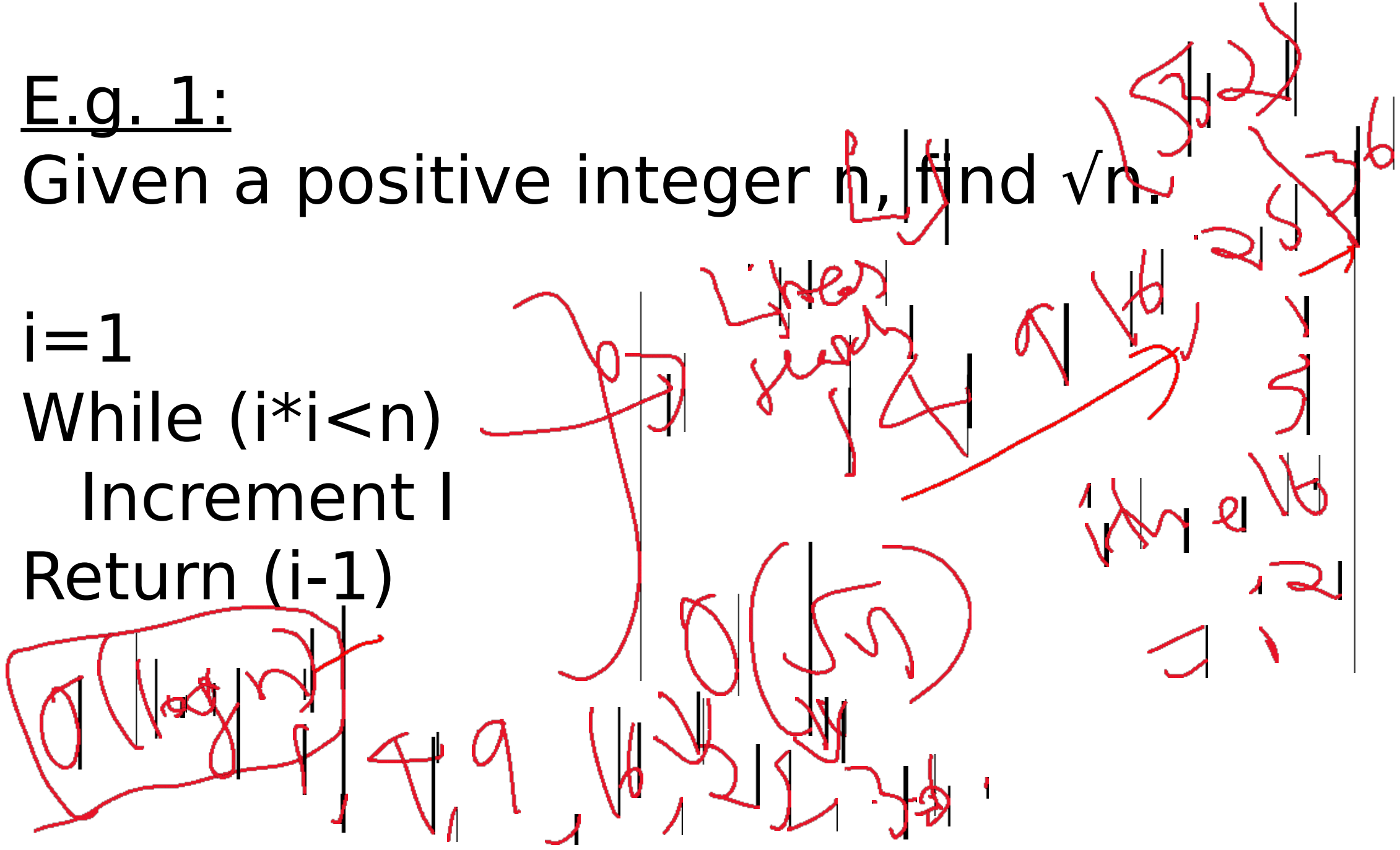




E.g. 1:

Given a positive integer n, find $\sqrt{n}$.

```
i=1
While (i*i<n)
   Increment I
Return (i-1)
```



E.x. 1:

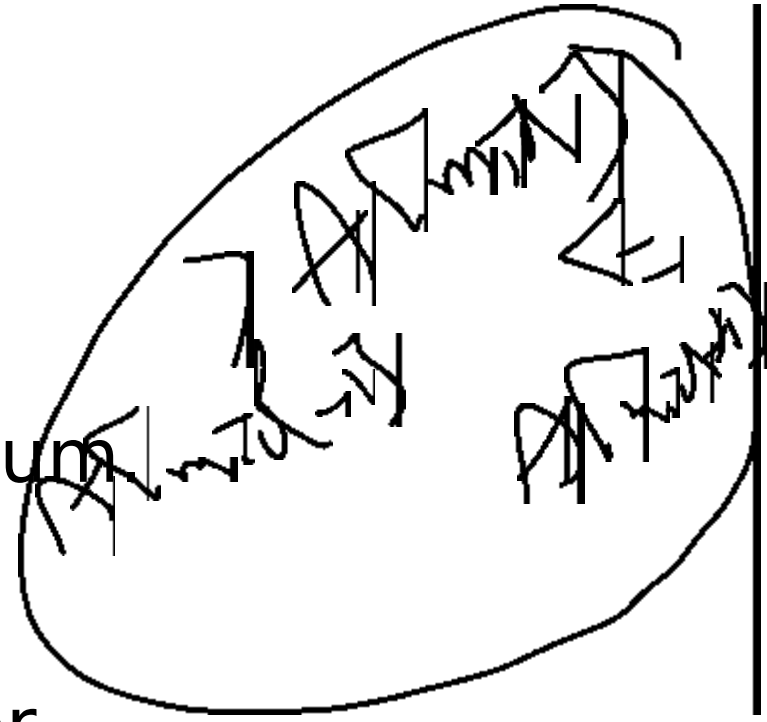
Finding a local minimum’

3,7,0,-2,12,8,10,1,4,2

A local minimum is an element A[i] such that it is:
A[i]<=A[i-1] and A[i]<=A[i+1]

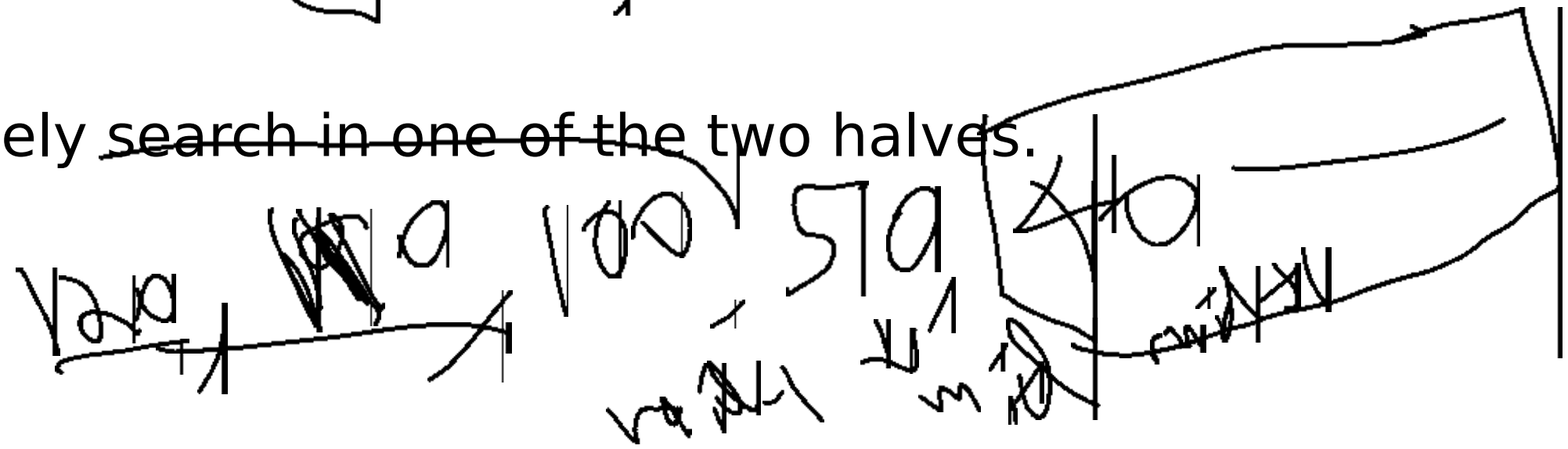
Algorithm for local minimum

1. Compare the middle element with its neighbors to check if it is a local minimum.
2. Suppose it is not.



A[mid] > A[mid-1] or A[mid]>A[mid+1] (or both)

Recursively search in one of the two halves.

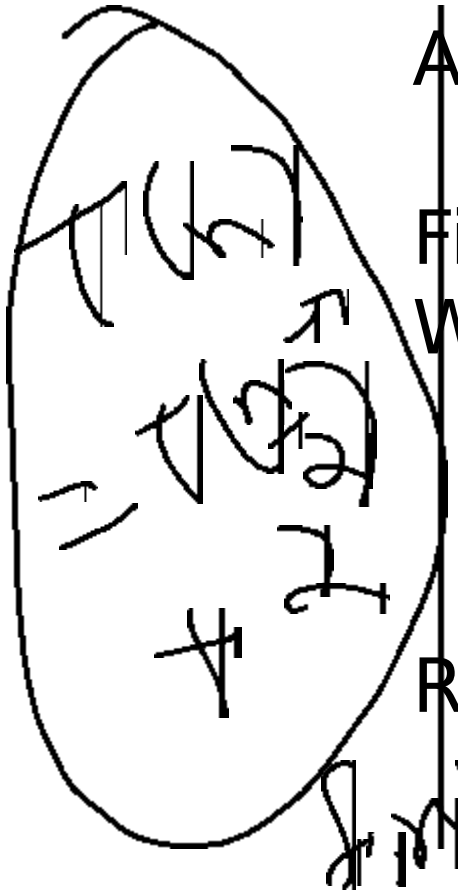


If A[mid+1]<A[mid], then the right half of the array contains at least one local minimum.

Proof:
The smallest element of the right half is a local minimum.

Algo to find local minimum

1. Compare middle element with its neighbors. If it is the local minimum, we are done.
2. Else if A[mid-1]<A[mid], then recursively search in A[first,mid-1]
3. Else (A[mid+1]<A[mid]), recursively search in A[mid+1,last]



Algo to find local minimum

First=1,last=n
While (first<last)
    If A[mid] is a local minimum, return mid.
    Else if A[mid-1]<A[mid], set last=mid-1
    Else if A[mid+1]<A[mid], set first=mid+1
Return first.

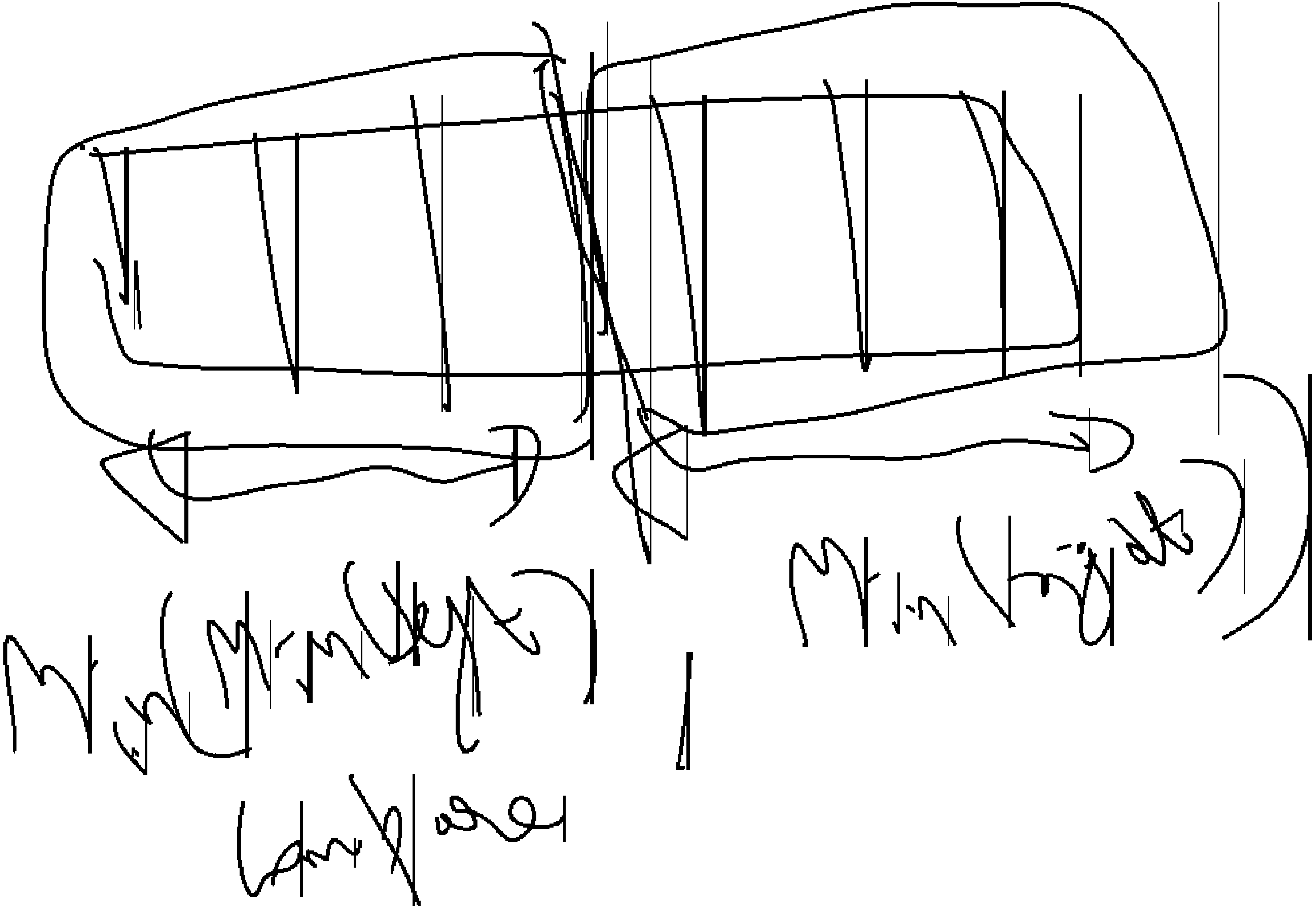
Algo for standard binary search: A[1]<=...<=A[n] and x

First=1,last=n
While (first<last)
    If (x=A[mid), return mid.
    Else if x<A[mid], set last=mid-1
    Else if x>A[mid], set first=mid+1
If (x=A[first]) return first, else return Not found.

Modify it to find the least index I such that A[i]<=x
2,4,6,8,10,12 and x=11. Output: 5

Comparisons in finding minimum: n − 1

Every comparison eliminates 1 elt

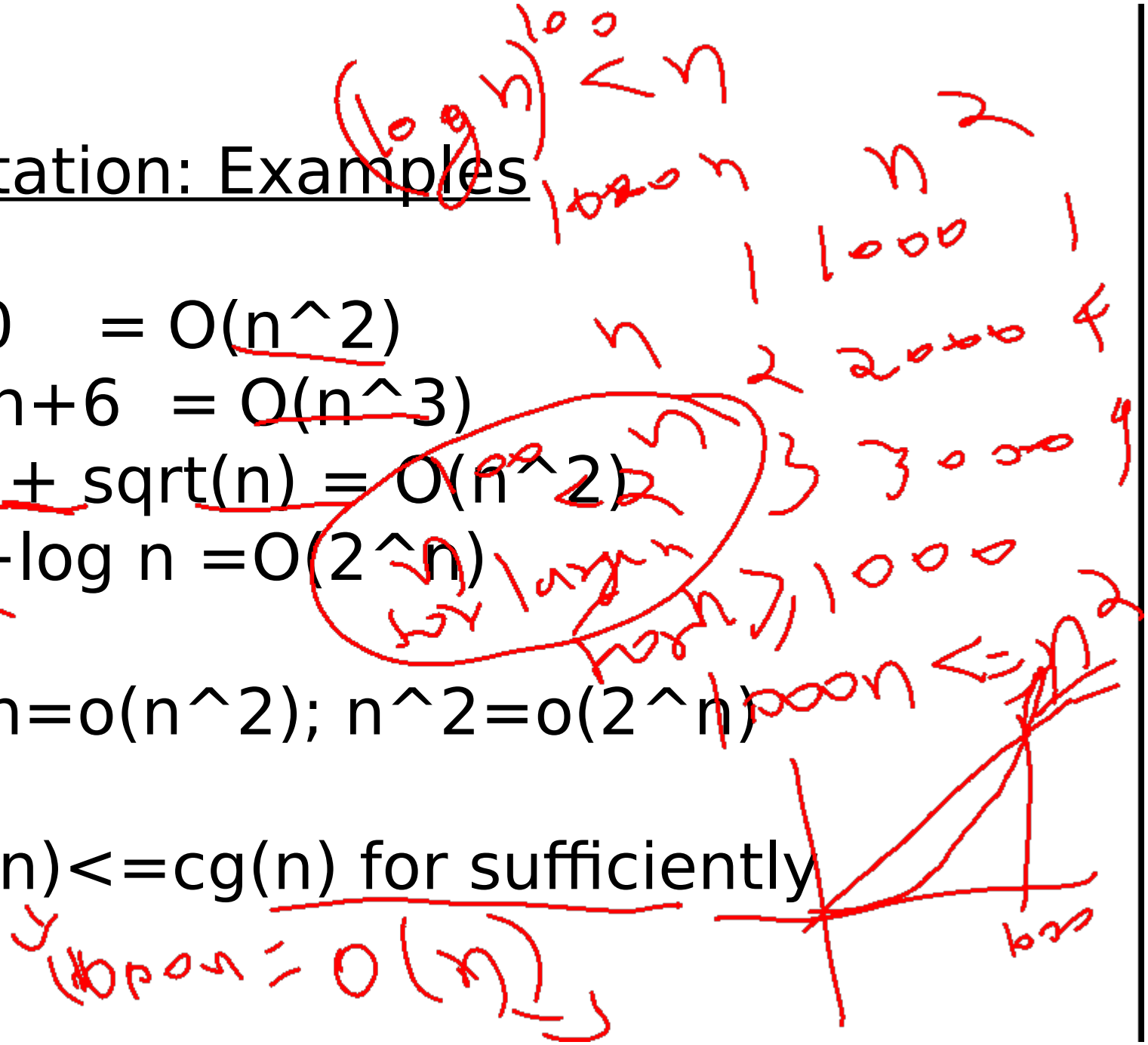


Asymptotic Notation: Examples

1. 4n^2+5n-10 = O(n^2)
2. n^3-n^2+5n+6 = O(n^3)
3. n^2+nlog n + sqrt(n) = O(n^2)
4. 2^n+n^10+log n =O(2^n)

5. log n=o(n); n=o(n^2); n^2=o(2^n)

F(n)=O(g(n)) f(n)<=cg(n) for sufficiently large n

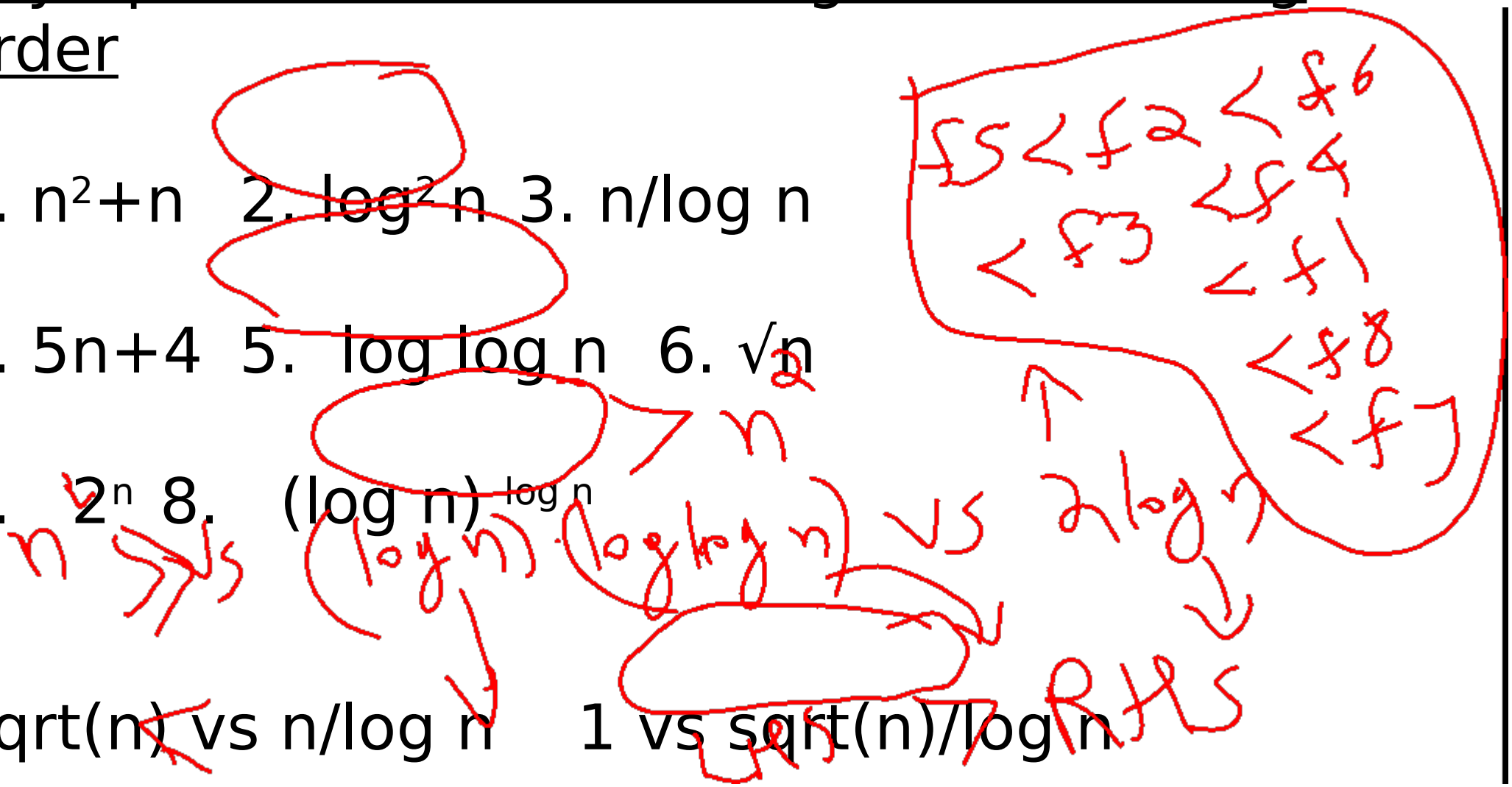


Asymptotic Notation: Arrange in increasing order

1. n^2+n 2. $\log^2 n$ 3. $n/\log n$

4. $5n+4$ 5. $\log \log n$ 6. $\sqrt{n}$

7. 2^n 8. $(\log n)^{\log n}$



Sqrt(n) vs n/log n 1 vs sqrt(n)/log n