CS218
Design and Analysis of Algorithms

Instructor: Rohit Gurjar

TAs : Roshan Raj, Dhiraj Kumar Shah, Prathamesh Yeole, Suraj Munjani,, Swapnil Bhattacharyya, Anuj Asati, Yash Sadhwan, Jaya Bharti

Logistics

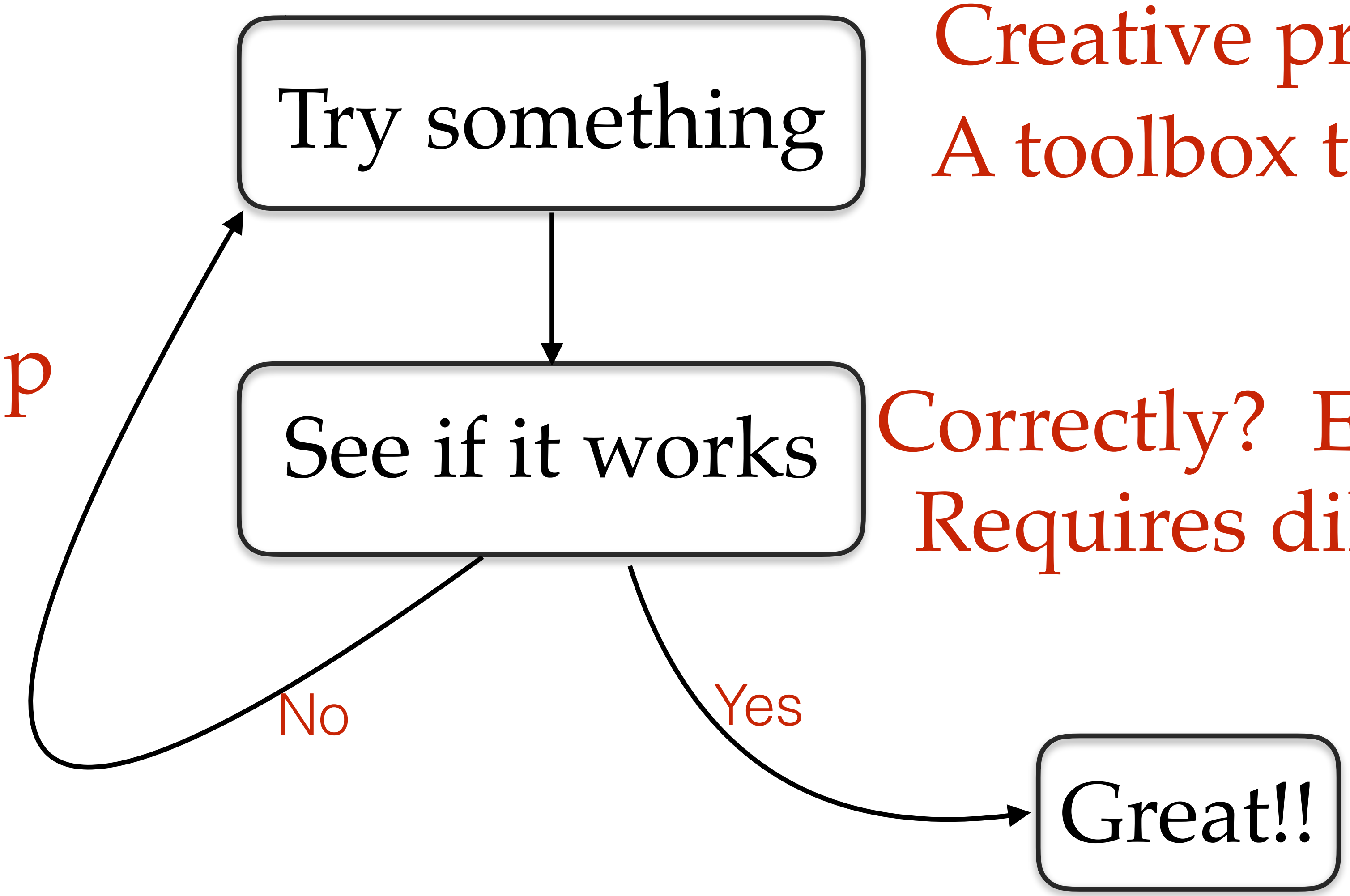
- **Class**: Mon 8:30, Tue 9:30, Thu 10:35. LA001.
- **Office hours**: Fri 4-5 pm. CC315
- Tutorials: will schedule
- Slides, exercises etc. on course webpage
  https://www.cse.iitb.ac.in/~rgurjar/CS218-2024/
- Announcements, doubts/discussions/exercises: Piazza.
  http://piazza.com/iit_bombay/spring2024/cs218
  Access code: cs218

Grading

- 3 Programming assignments - 5+5+5 %
- 2 quizzes- 10+10% (Jan 31, Mar 27, 8:30-9:30 AM)
- Midsem - 25 %
- Endsem - 40 %

Objectives

- How to design algorithms.



Try something

See if it works

No

Yes

Great!!

Creative process
A toolbox to help

When to stop
this loop?

Correctly? Efficiently?
Requires diligence

Algorithm's correctness

- How to argue that an algorithm is **not** correct: show bad examples
- Arguing algorithm's correctness
  - ~~**Correctness by confidence**: I came up with it, so it must be correct~~
  - ~~**Correctness by examples**: because it works for my f examples~~
  - ~~**Correctness by authority**: It's just obvious~~
- Formal proofs of correctness, whenever needed
- How to communicate/represent an algorithm

What will you learn

- **Principles of designing and analyzing algorithms:**
    - Basic principles like induction/recursion.
    - Divide and Conquer,
    - Dynamic Programming,
    - Greedy Algorithms.
- **Beyond the basics:**
    - Bipartite Matching
    - Network Flow and applications.
    - Reductions.
- **Complexity:**
    - Polynomial time and the Complexity classes NP, co-NP.
    - NP-hardness.
- **Advanced topics:**
    - Randomized algorithms
    - Approximation algorithms
- **Miscellaneous:**
    - How QR codes work?

References

- Kleinberg, Tardos (amazing book, freely available)
- Prof. Sundar’s course notes

Binary Search and Variants

- Applicable whenever it is possible to reduce the search space by half using one query

- Search space size N
  number of queries = $O(\mathit{log}\ N)$

Classic example

- Given a sorted array A of integers, find the location of a target number x (or say that it is not present)

Pseudocode:

```
Initialize start $\leftarrow$ 1, end $\leftarrow$ $n$;
Locate($x$, start, end){
    if (end < start) return not found;
    mid $\leftarrow$ (start+end)/2;
    if ($A$[mid] = $x$) return mid;
    if ($A$[mid] < $x$) return Locate($x$, mid+1, end);
    if ($A$[mid] > $x$) return Locate($x$, start, mid-1);
}
```

Other examples

- Looking for a word in the dictionary
- Debugging a linear piece of code
- Cooking rice
- Science/Engineering: Finding the right value of any resource
    - length of youtube ads
    - pricing of a service

Egg drop problem

- In a building with n floors,
  find the highest floor from where egg can be dropped without breaking.

- How many egg drops do we need?

- $O(\log n)$ egg drops are sufficient.

Egg drop: unknown range

- In a building with *infinite* floors,
  find the highest floor *h* from where egg can be dropped without breaking.

- *$O(\log h)$* egg drops sufficient?

Lower bound

- Search space size: *N*
  Is *$O(\log N)$* queries necessary?

Exercise: subarray sum

- Given an array with n positive integers,
  and a number S,
  find the minimum length array whose sum is S?

- Subarray is a contiguous subset, i.e.,
  A[i], A[i+1], A[i+2], …, A[j-1], A[j]

- Trivial: $O(n^2)$
  Check all subarrays

- Can we do this in $O(n \log n)$?

Exercise: subarray sum

- Given an array with n positive integers,
  and a number S,
  find the minimum length array whose sum is S?

Exercises

- Given two sorted arrays of size n, find the median of the union of the two arrays.
  $O(log\ n)$ time?

- Given a convex function $f(x)$, find integer x which minimizes $f(x)$.

- Land redistribution:
  given list of landholdings $a_1, a_2, \ldots, a_n$,
  given a floor value f,
  find the right ceiling value c

Division algorithm

Finding square root

- Given integer a, find

Inverse of a monotone function