How to Improve Implementation

- Gaurish Baliga

Goals:

1. Solve Problems fast
2. Solve Harder Problems

] Quality of Implement..

Implementation Skills

2022

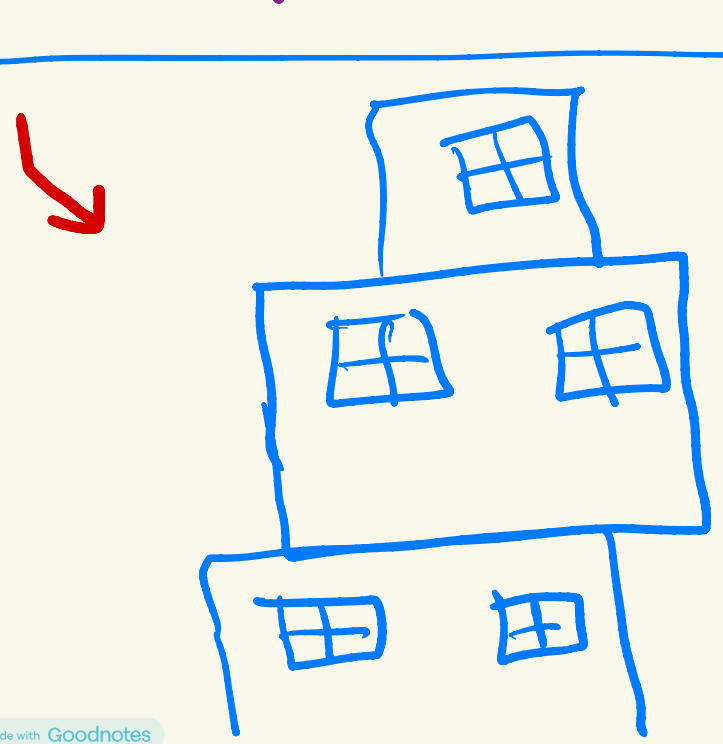
```
void solve(){
  int n;
  cin >> n;

  vint answer(n);
  cin >> answer;

  vint prefix(n,0);
  vint suffix(n,0);
  for(int i=1;i<n-1;i++){
    prefix[i] = max(0ll,max(answer[i-1],answer[i+1])-answer[i]+1);
    if(i>=2)prefix[i]+=prefix[i-2];
  }

  for(int i=n-2;i>0;i--){
    suffix[i] = max(0ll,max(answer[i-1],answer[i+1])-answer[i]+1);
    if(i<=n-3)suffix[i]+=suffix[i+2];
  }
  int ans = 1e18;
  if(n&1){cout<<prefix[n-2]<<"\n";return;}
  for(int i=1;i<n-1;i+=2){
    int temp = 0;
    if(i>0)temp+=prefix[i];
    if(i==n-4)temp+=suffix[i+2];
    else if(i<n-4)temp+=min(suffix[i+2],suffix[i+3]);
    ans = min(temp,ans);
  }
  ans = min(ans,suffix[2]);
  cout<<ans<<"\n";



}
```

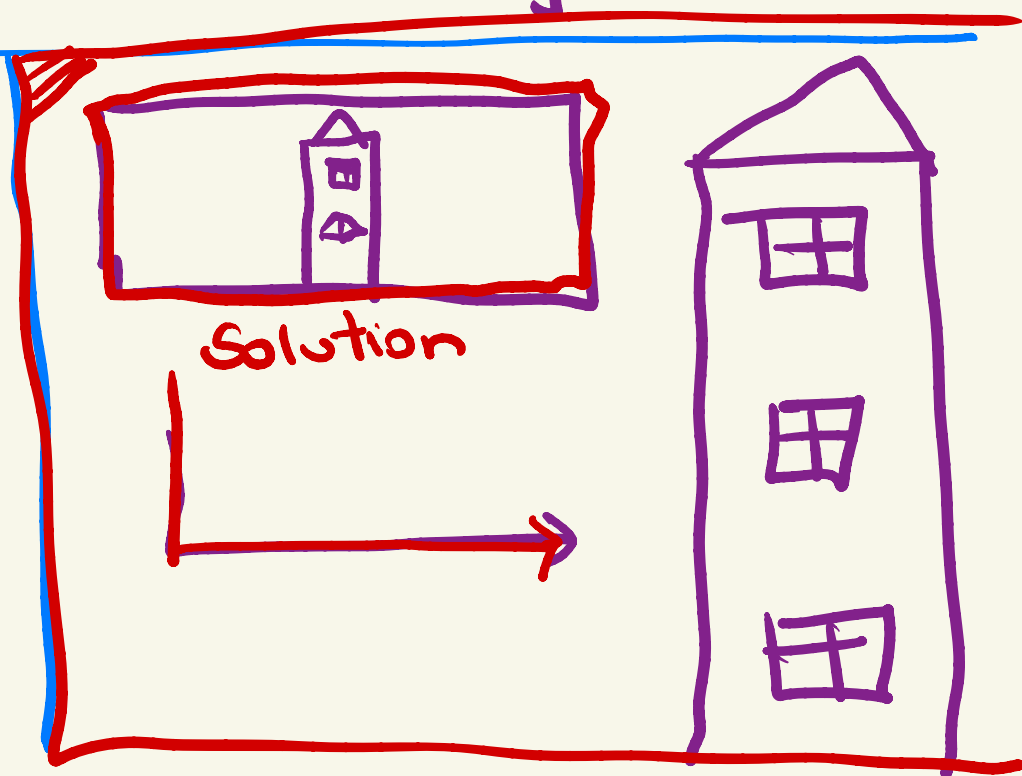
Cleaner

Readable

```
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif

    int tt; cin >> tt;

    while(tt--) {
        int n; cin >> n;

        vector<int>a(n), prefix(n), suffix(n);
        for(auto &i : a) cin >> i;

        for(int i = 1; i + 1 < n; i++) {
            prefix[i] = max(0ll, max(a[i - 1], a[i + 1]) - a[i] + 1);
            if(i != 1) prefix[i] += prefix[i - 2];
        }

        for(int i = n - 2; i >= 1; i--) {
            suffix[i] = max(0ll, max(a[i - 1], a[i + 1]) - a[i] + 1);
            if(i != n - 2) suffix[i] += suffix[i + 2];
        }

        if(n % 2 == 1) cout << prefix[n - 2] << "\n";
        else {
            int answer = suffix[2];
            for(int i = 1; i + 1 < n; i += 2) {
                answer = min(answer, prefix[i] + (i + 3 < n ? suffix[i + 3] : 0ll));
            }
            cout << answer << "\n";
        }
    }
}
```

1. Come up w/ entire solution & then start coding.





2. Standard Techniques

How??

↳ 1 to $2 \cdot 10^5$

↳ i

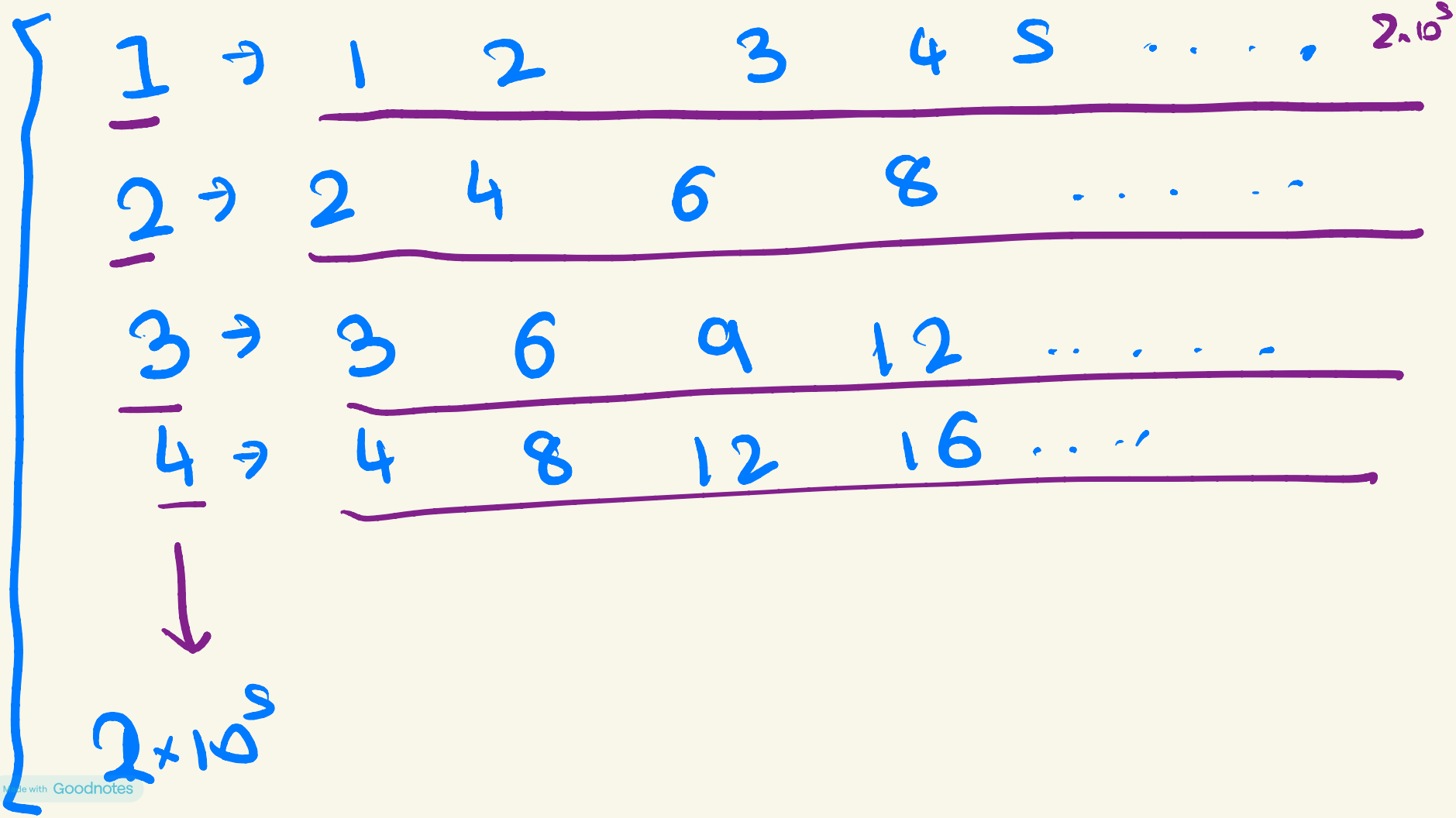
↳ all multiples of

$i \leq 2 \times 10^5$

$2 \rightarrow 2 \times 10^5$

$i \times 2 \times 10^5$

$m \leq 2 \times 10^5$



```
for (i = 1 ; i <= 2*10^5 ; i++) {
    for (j = i ; j <= 2*10^5 ; j++) {
        if (j % i == 0) {
            print(j)
```

multiple

}
}
}

$O(n^2)$

i → i 2i 3i → 3i+1 (4i)

concept

```
for(i=1; i <= 2*10^5; i++) {
    for (j=i; j <= 2*10^5; j+=i) {
        print(j);
    }
}
```

* Multiples of i

O(n log n)

~~O(n^2)~~

$i = 1 \rightarrow \frac{n}{1}$

$i = 2 \rightarrow \frac{n}{2}$

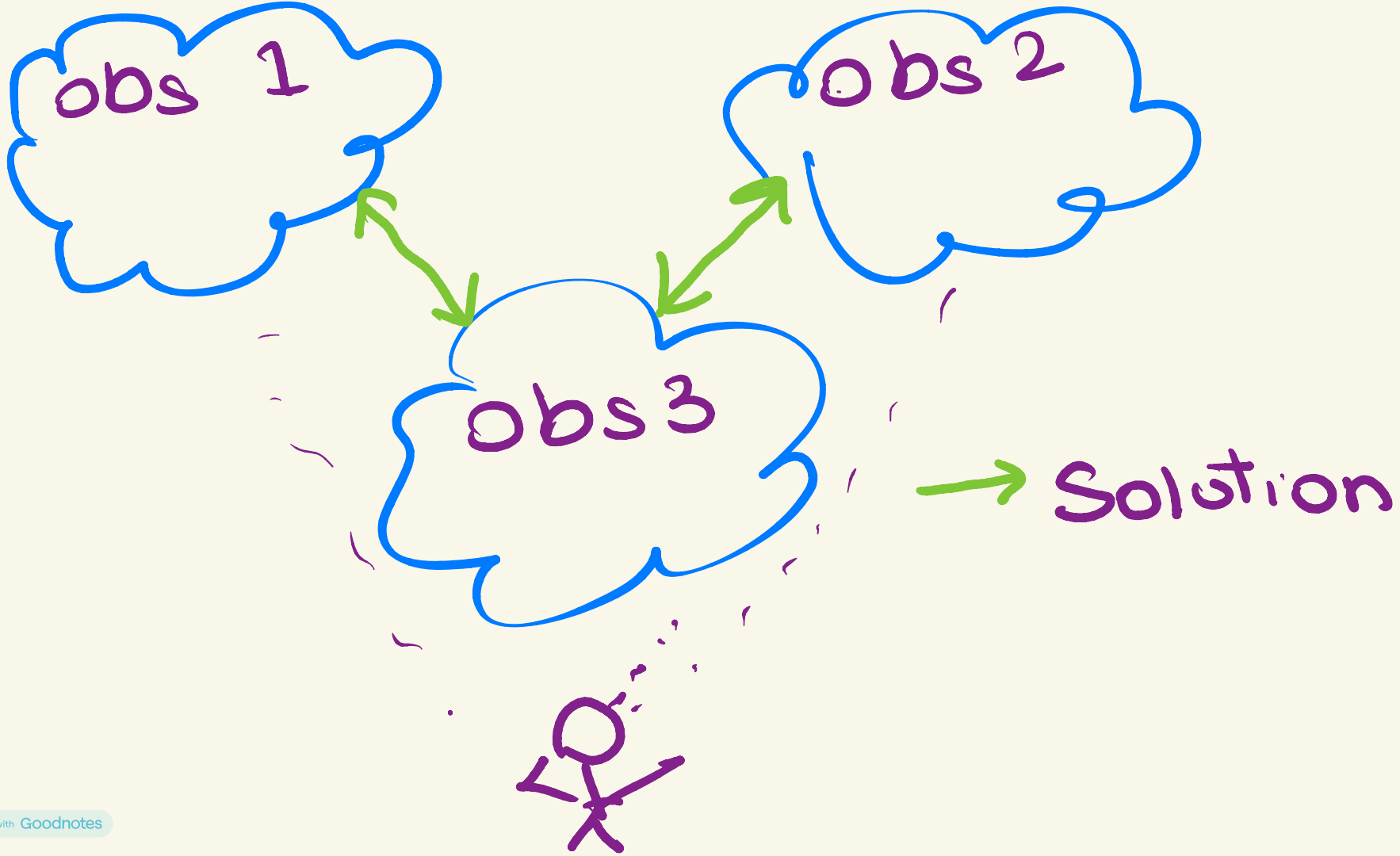
$i = 3 \rightarrow \frac{n}{3}$

$\sum_{i=1}^{n} \frac{n}{i}$ HP

Harmonic Series

$\frac{n}{1} + \frac{n}{2} + \frac{n}{3} \cdots \frac{n}{n}$

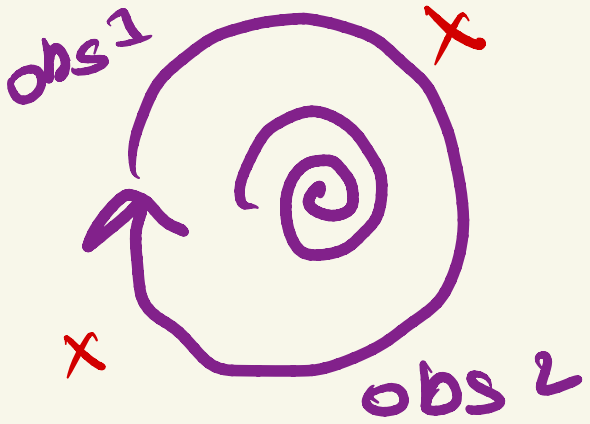
$n \log n$



~~# obs 1~~

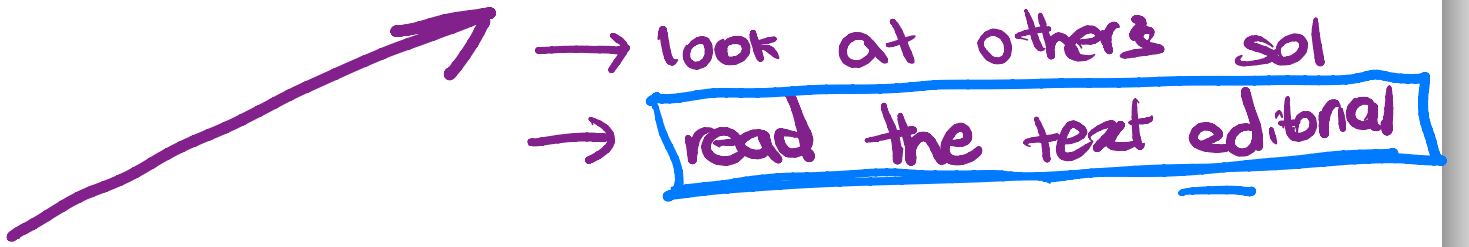
obs 2

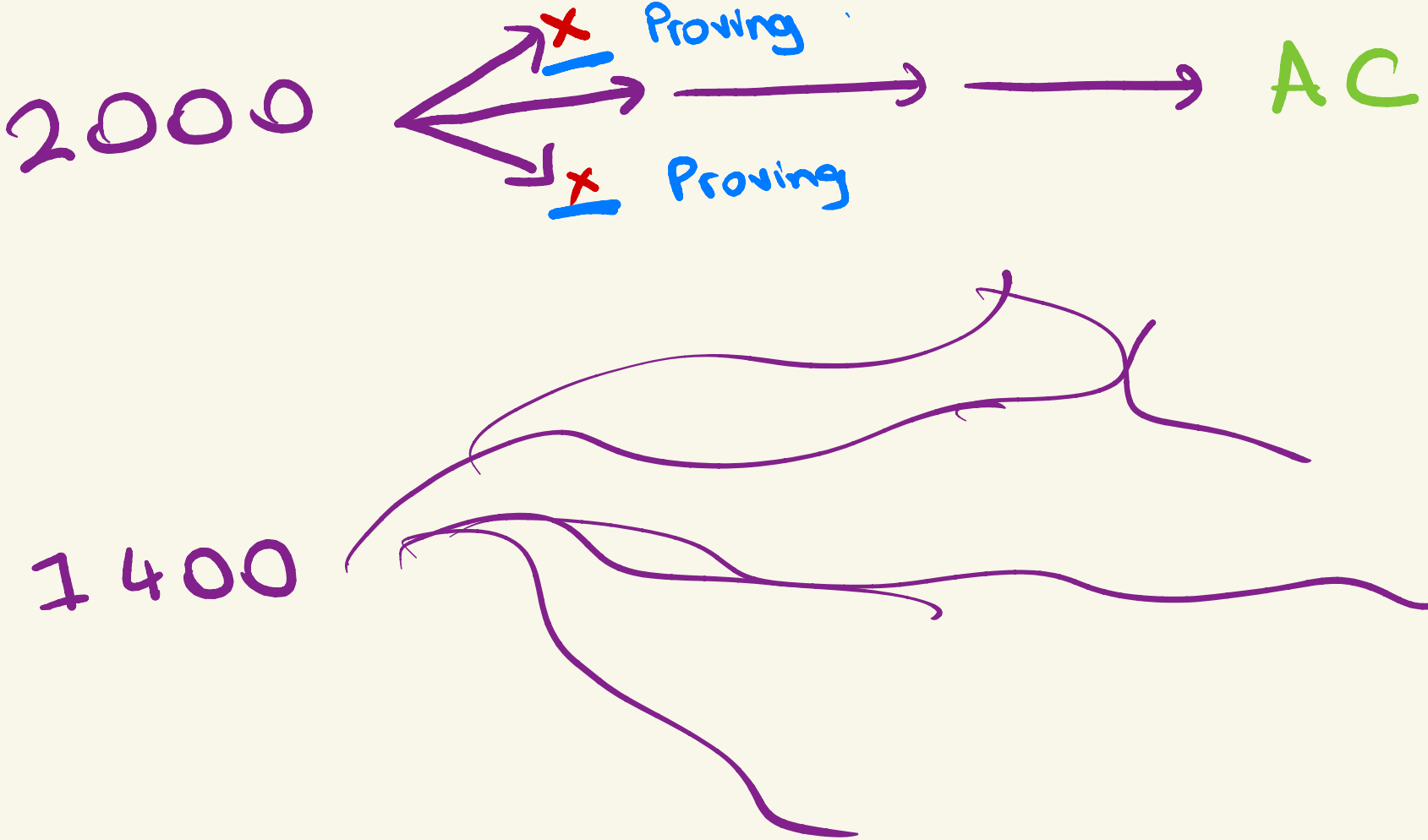
Thoughts



How to Improve Implementation Skills?

- Come up with solution before starting to code ✓
- Keep pen and paper alongside you and clarify all testcases ✓
- During practice, once you have implemented your accepted verdict, look at the code of people who write clean code and see how they have implemented the same thing





General Implementation Tricks

Trick #1

- Smaller Implementation of vector

3D vector

```
int n = 10, m = 10, k = 10;

vector<vector<vector<int>>>a(n, vector<vector<int>>(m, vector<int>(k)));

vector b(n, vector(m, vector(k, 0)));
```

OLL

int

C++ 20

Pythonish

'c'

Optional

Jiangly

Lambda Functions (Using Functions)

int main() {

```cpp
int n; cin >> n;

vector<int>a(n);
for(auto &i : a) cin >> i;

for(int i = 1; i < n; i++) a[i] += a[i - 1];

auto query = [&](int l, int r) -> int {
    return (a[r] - a[l - 1]) * (a[r] - a[l - 1]);
};

int q; cin >> q;

for(int i = 0; i < q; i++) {
    int l, r; cin >> l >> r;
    cout << query(l, r) << "\n";
}
```

}

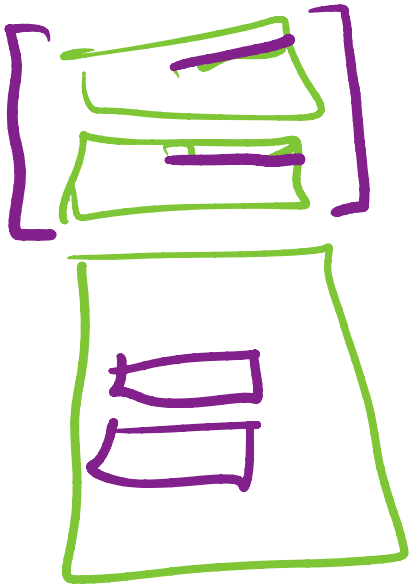
C++

name

param

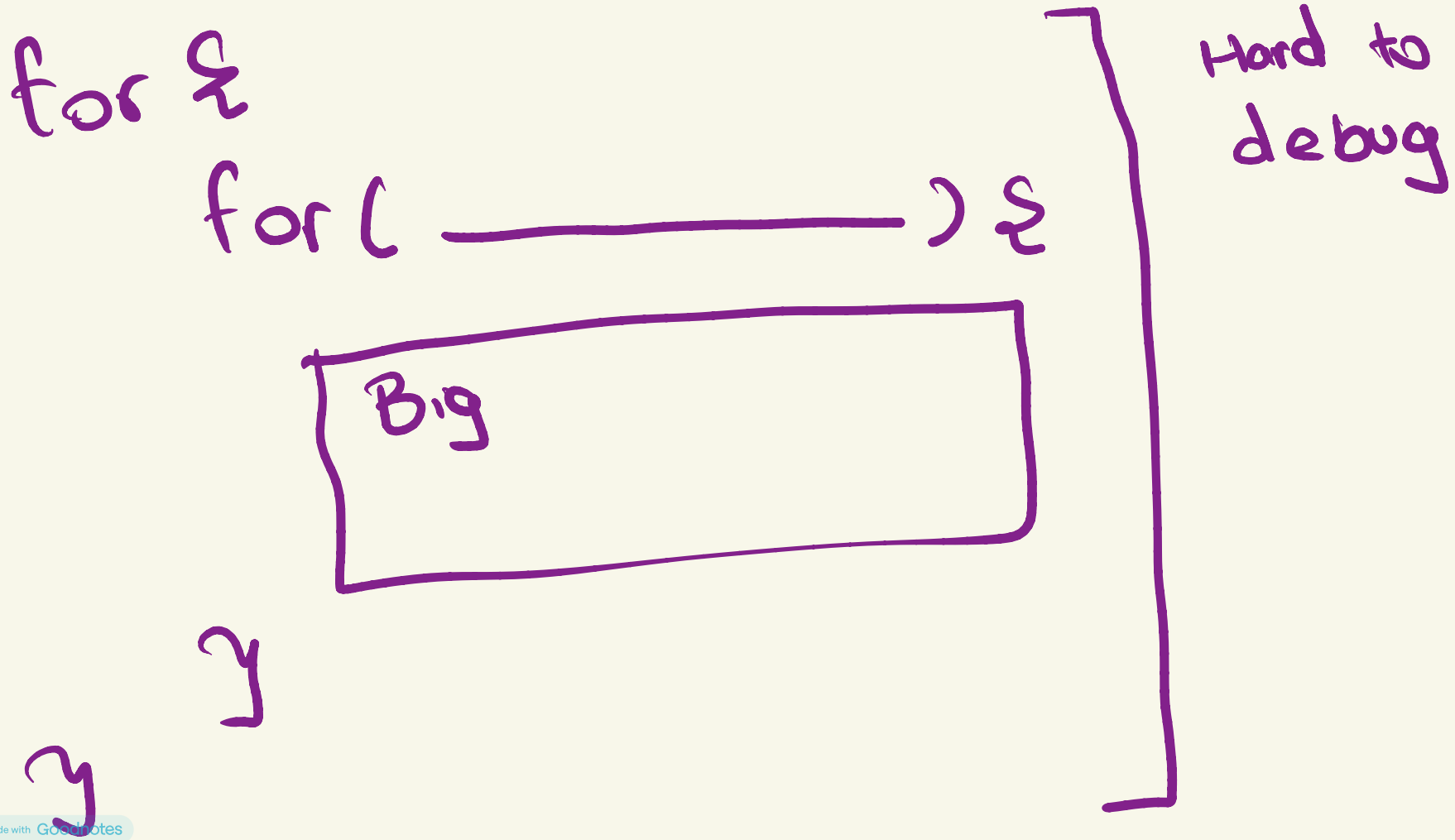
#

Prefix Sums



auto n = 1;

↳ deduces the data type



Recursive Lambda Functions

```
auto factorial = [&](int n, auto &&factorial) -> int {
    if(n == 0) return 1;
    return n * factorial(n - 1, factorial);
};

cout << factorial(5, factorial);
```

Advantages of Lambda Functions

- No need of passing many parameters to functions
- Ability to focus on main implementation first and then write the logic for each function (top down method)
- Less bug prone code since each functionality is in its own lambda function.
- Flow of code is easier to understand.

Using Structs

own data type

```cpp
struct person {
    int age, salary, isMarried;
}

int main() {
    int n; cin >> n;

    vector<person>a(n);

    for(int i = 0; i < n; i++) {
        cin >> a[i].age >> a[i].salary >> a[i].isMarried;
    }
}
```

much more readable

Profiles for Clean Implementation

1. https://codeforces.com/profile/jiangly ✓
2. https://codeforces.com/profile/Queue → advocate of clean code

Was this session good?!

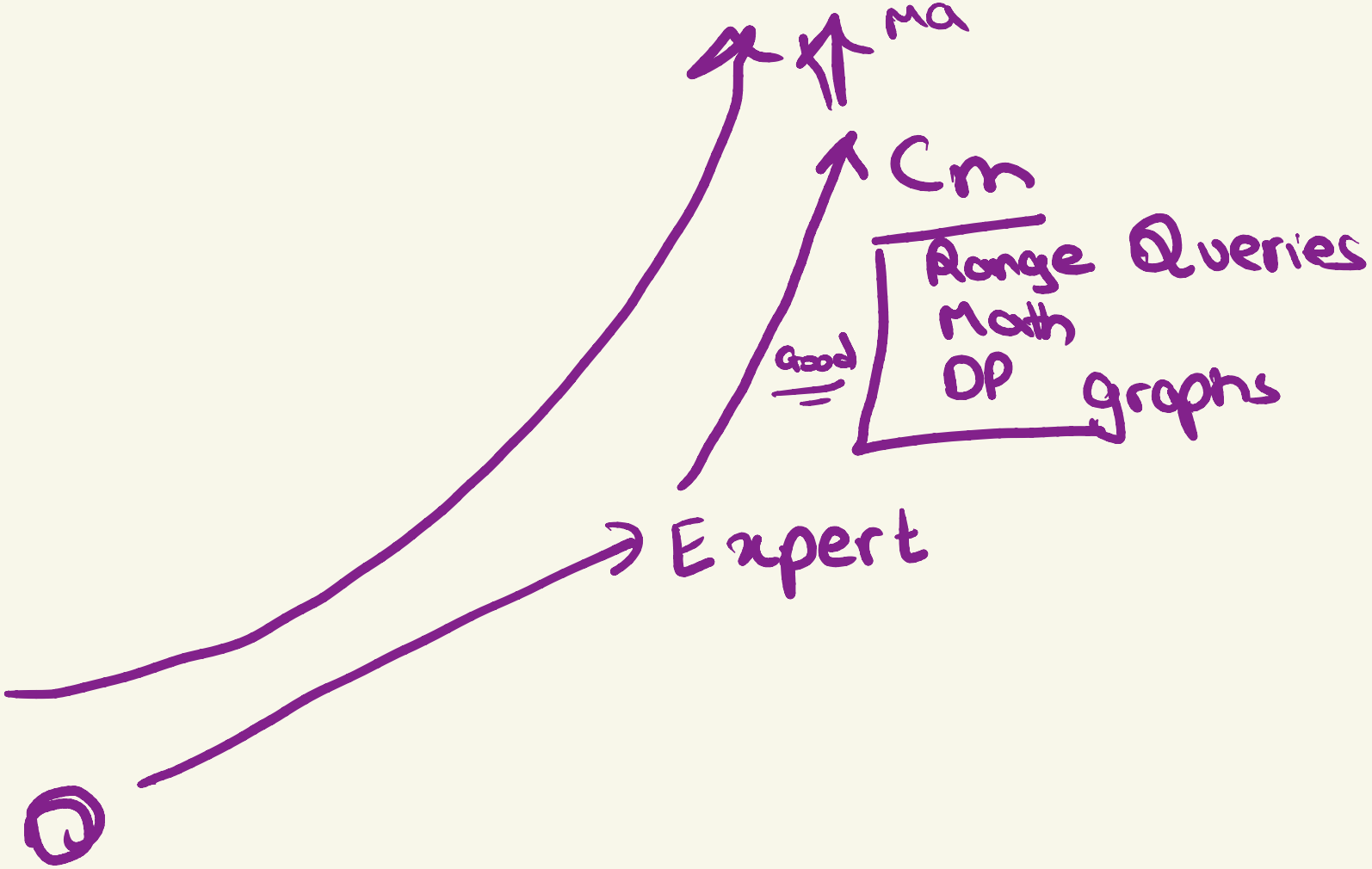
1st year → Only CP

↳ 4+ hours

↳ 4 months

Expert → [≈1700 rating problems around 30-40 min

CSES → searching sorting
→ graphs/dp/binary search]



New topic

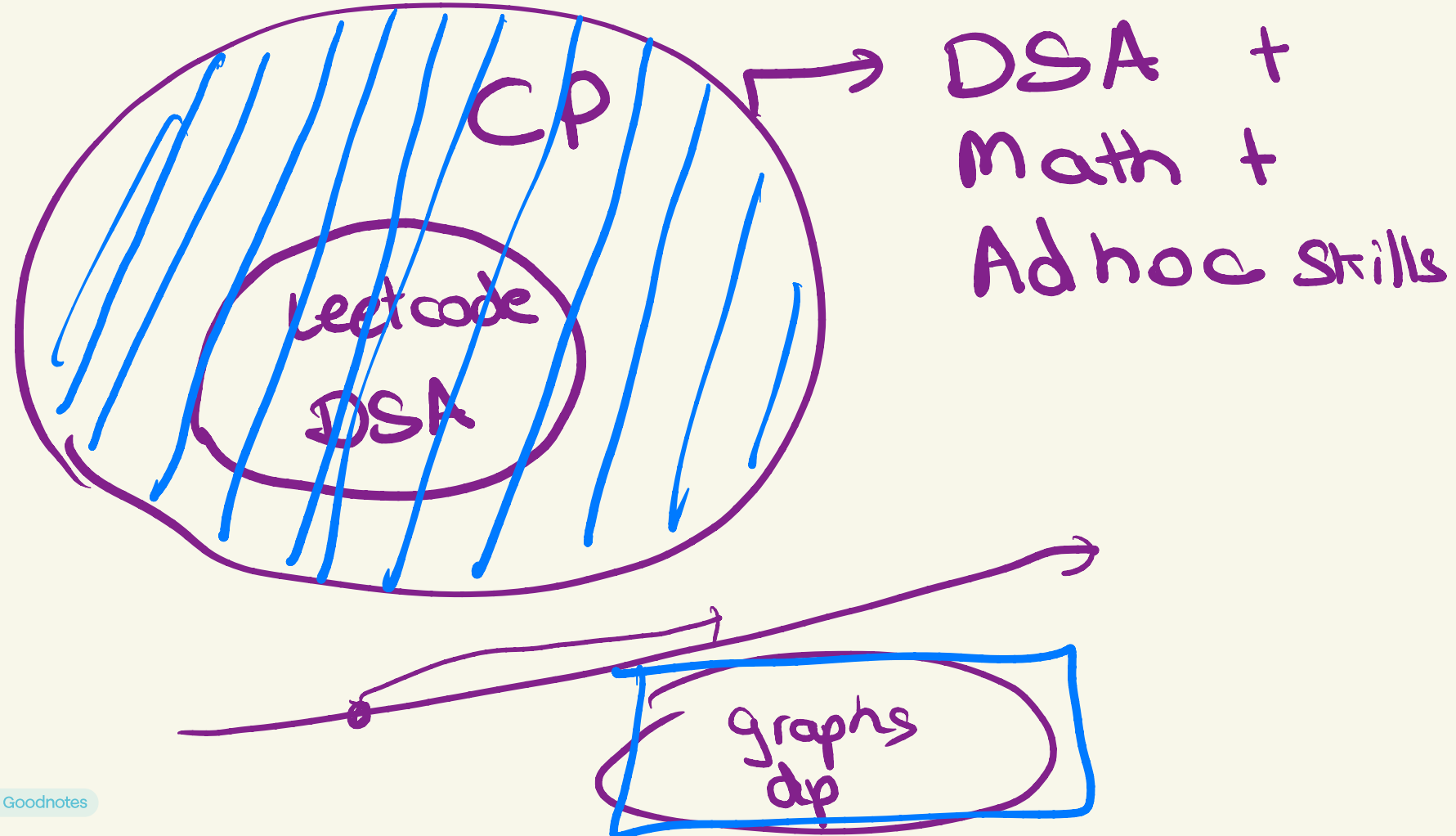
→ DPP → couple → very easily
→ mindsolve → skip

→ Solving by rating 2100

so random → rating X
↳ X + 100

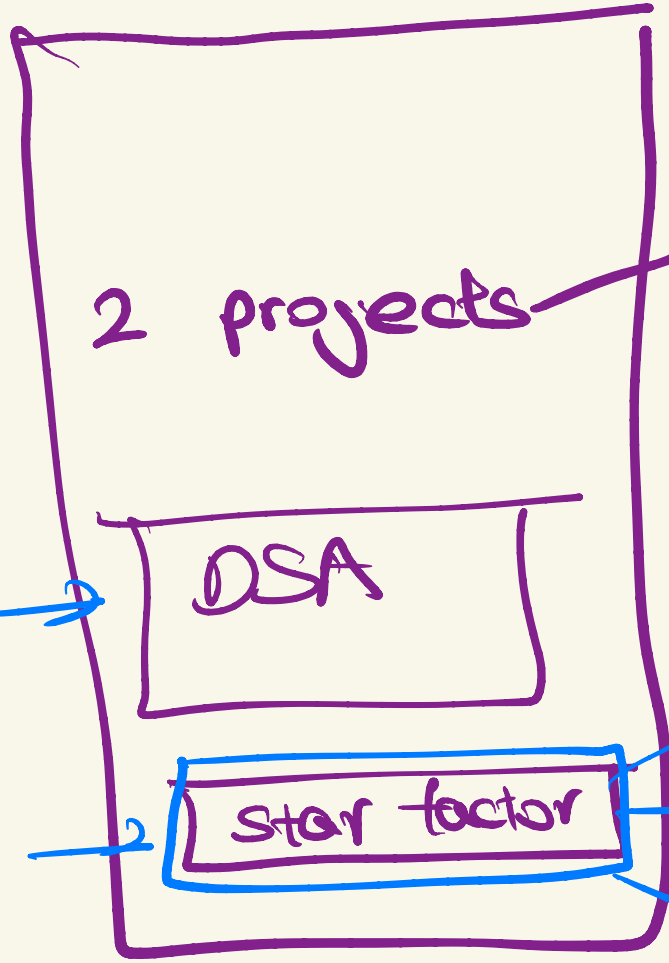
You should REALLY REALLY want to improve!!!

This is not JEE



Intern ✓✓

Good Resume



≥ 1700 skill

↳ many / most OA

≥ 1900 ⟶ almost all OAs