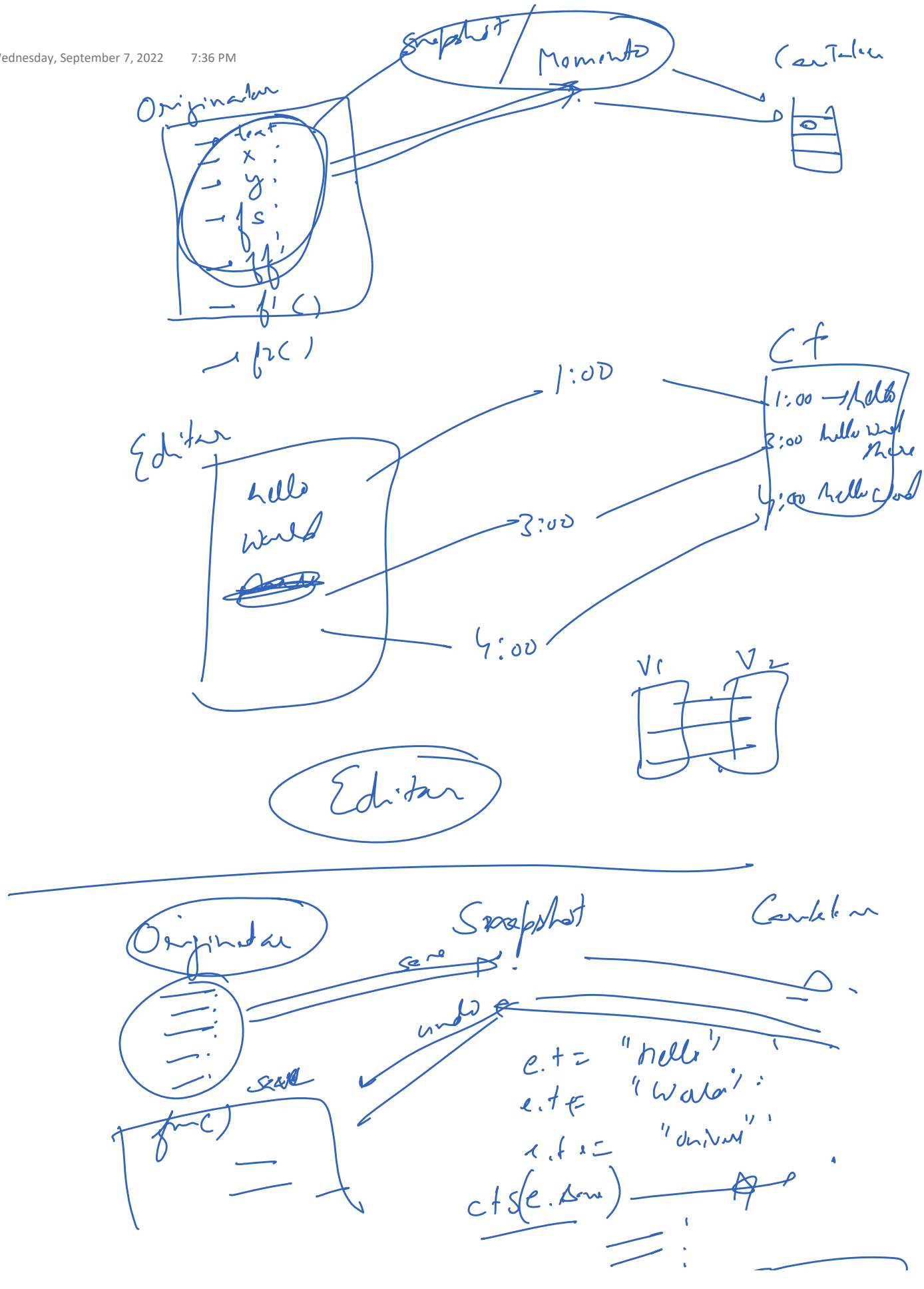
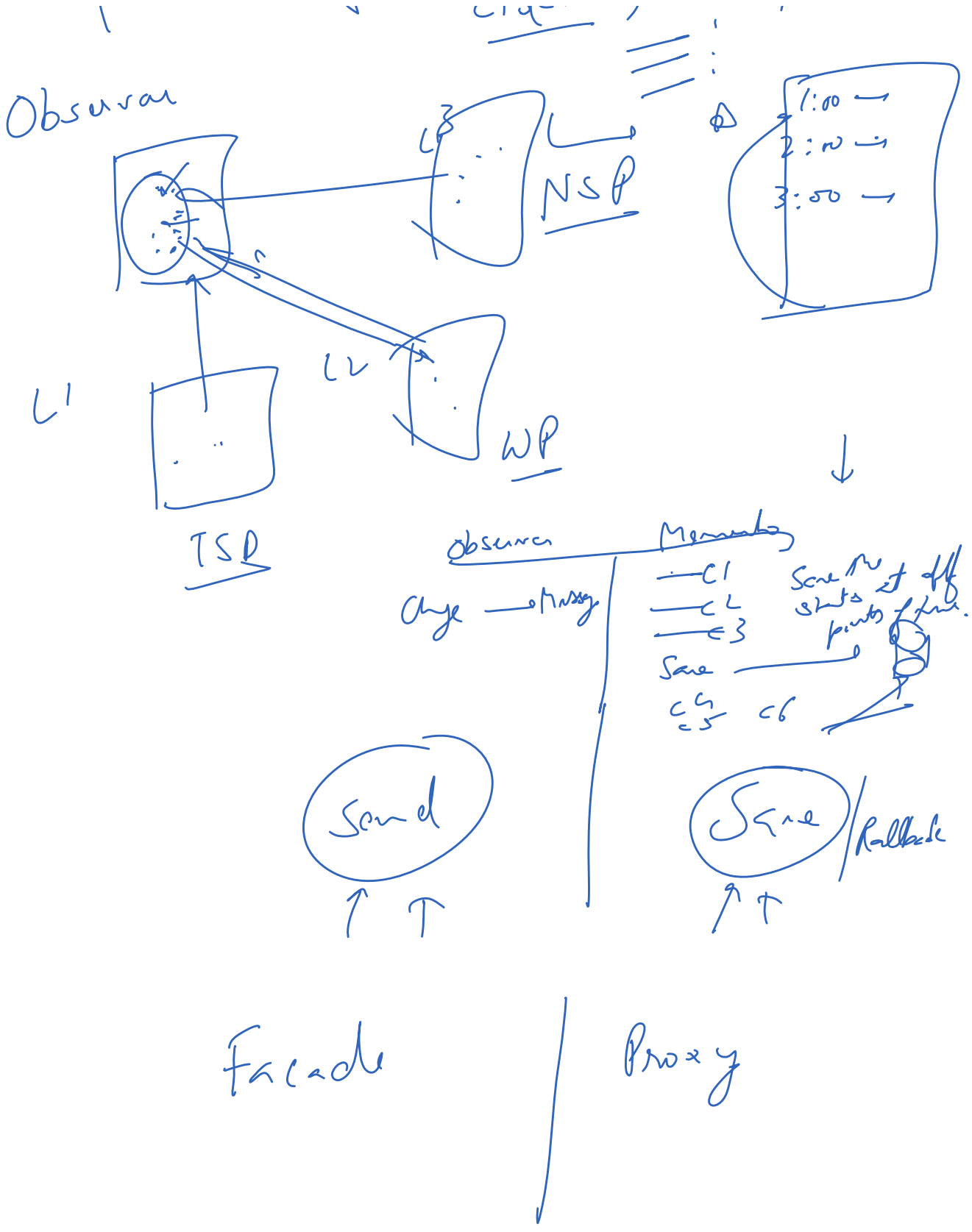

Originator
Memento
Caretaker
Editor
Hello
World
1:00
3:00
4:00
Editor
Originator
Snapshot
save
undo
save
func()
e.t = "Hello"
e.t = "World"
e.t = "Universe"
cts(e.time)


Observer
NSP
1:00
2:00
3:50
L1
L2
WP
TSP
Observer
Message
C1
C2
C3
Save
C4
C5
C6
Send
Save
Rollback
Facade
Proxy

Proxy

Ist

RC

PC

f1()

f1()

Facade = Boar

facade

A

f1()

f2()

work()

B

f1()

f2()

C

f1()

f2()

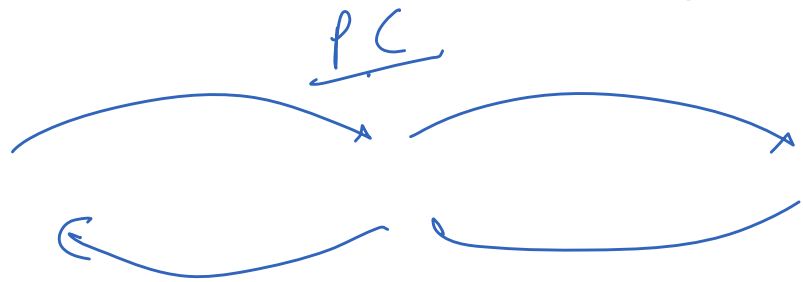
f3()

Client

Proxy

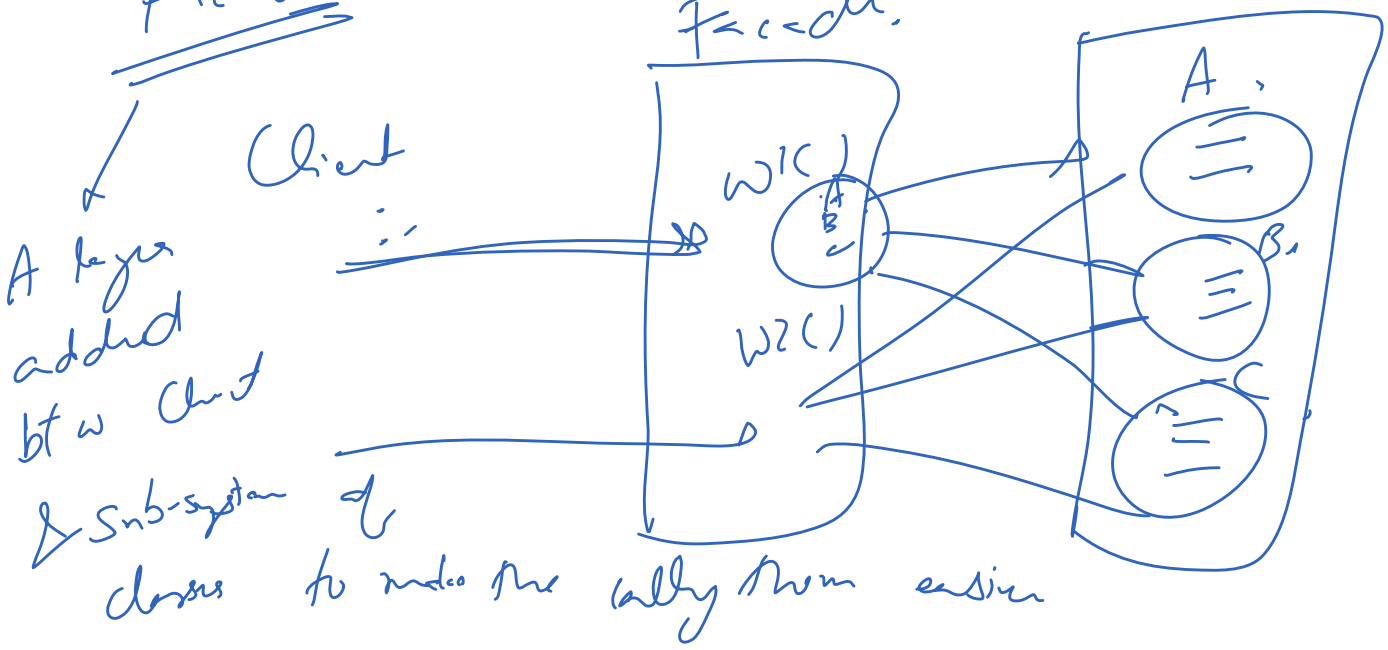
Client

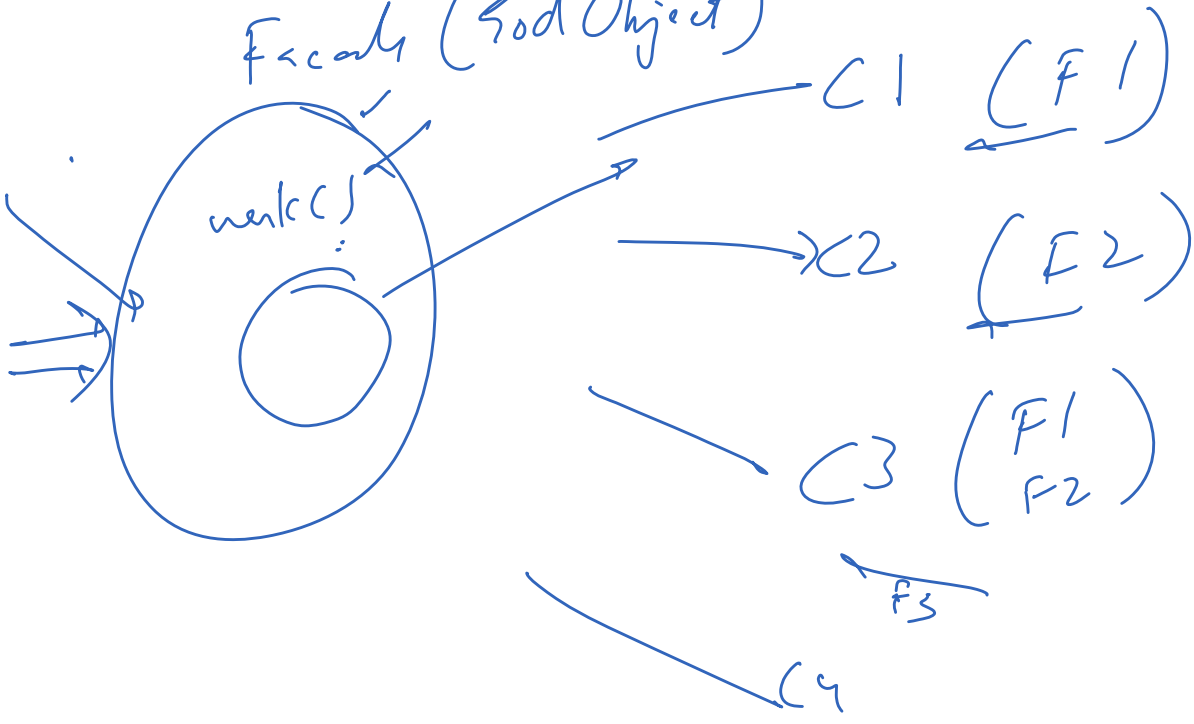
RC

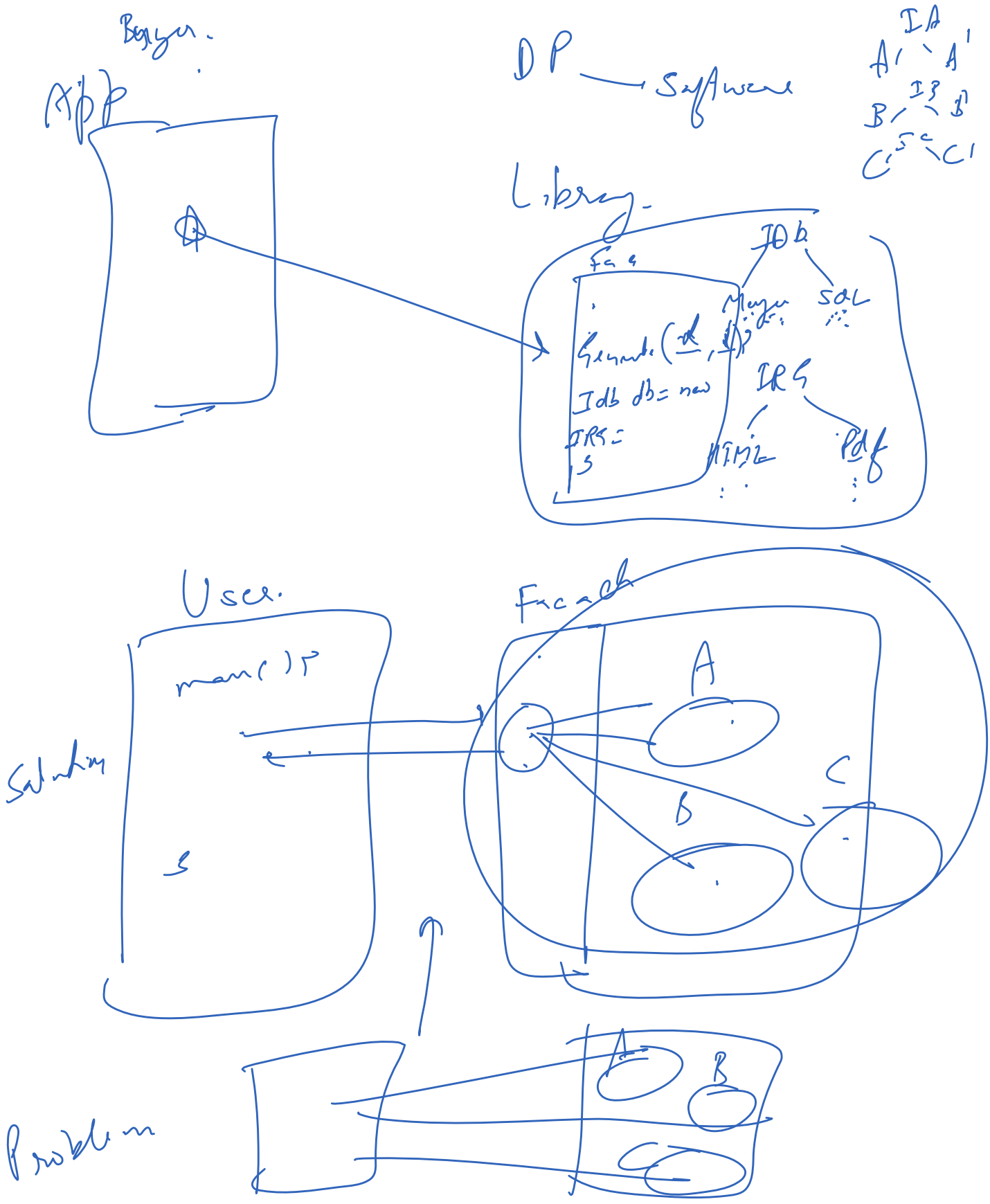


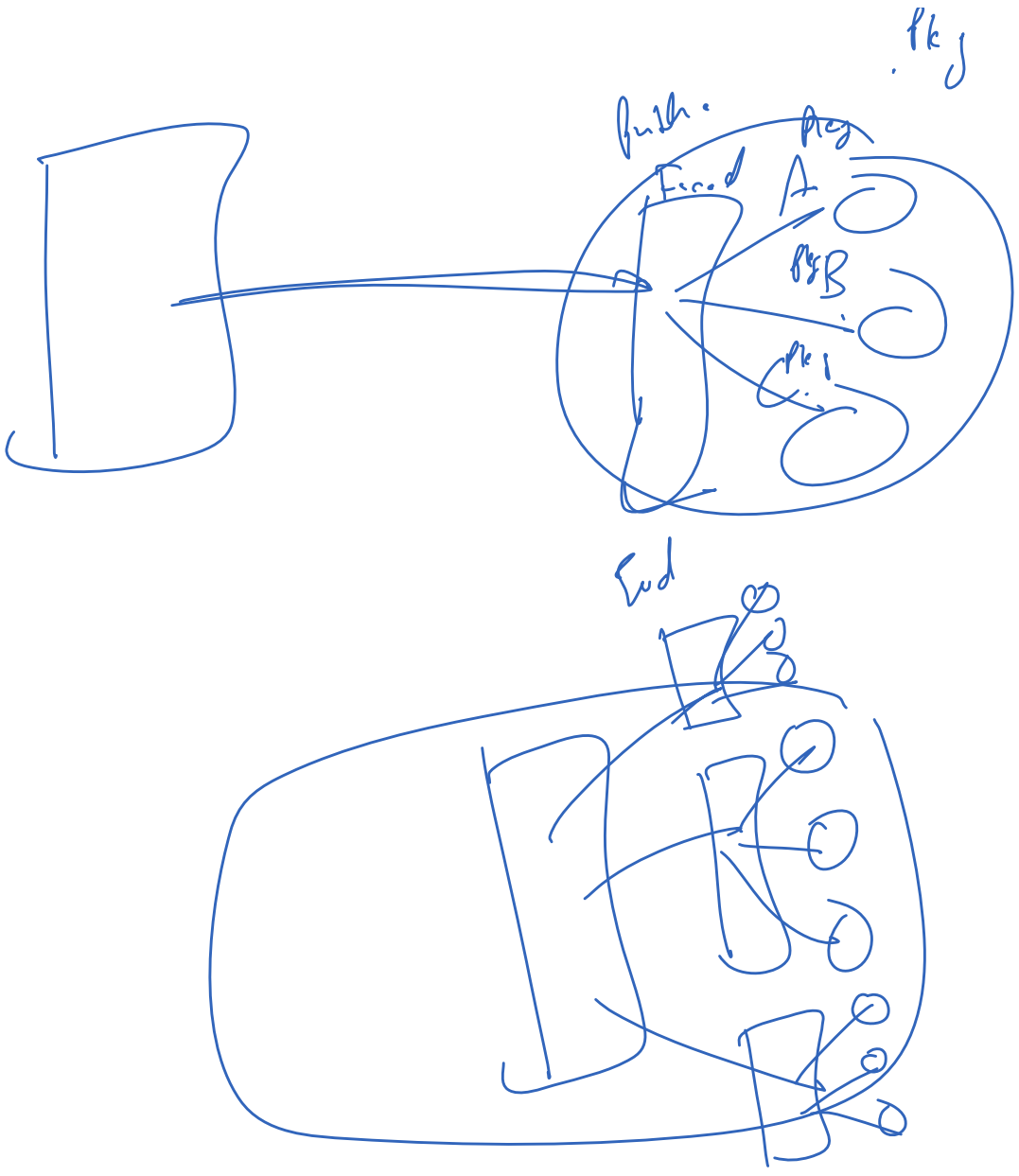
Facade

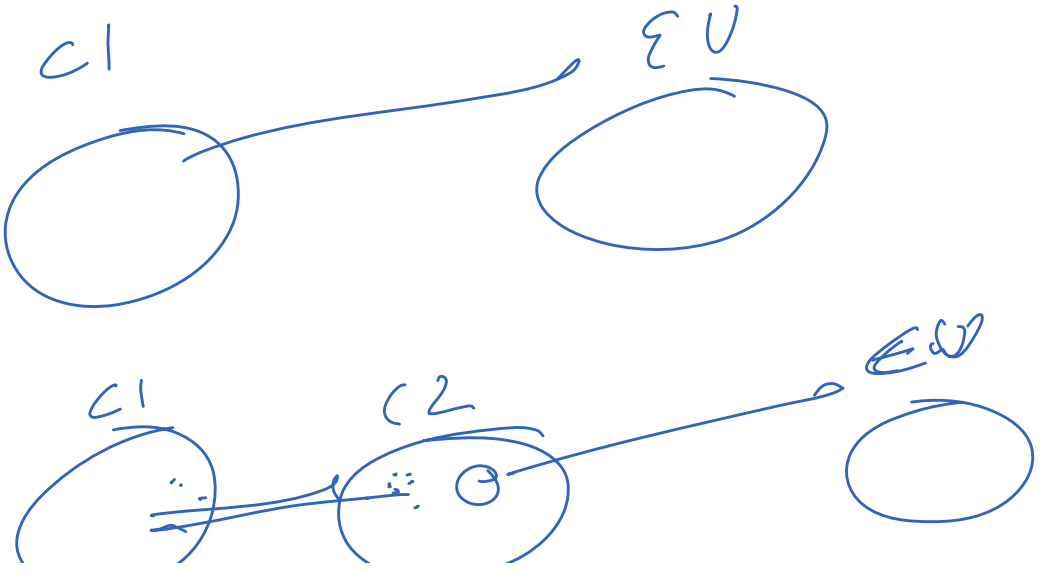
A layer added btw Client & Sub-system of classes to make the calls from easier

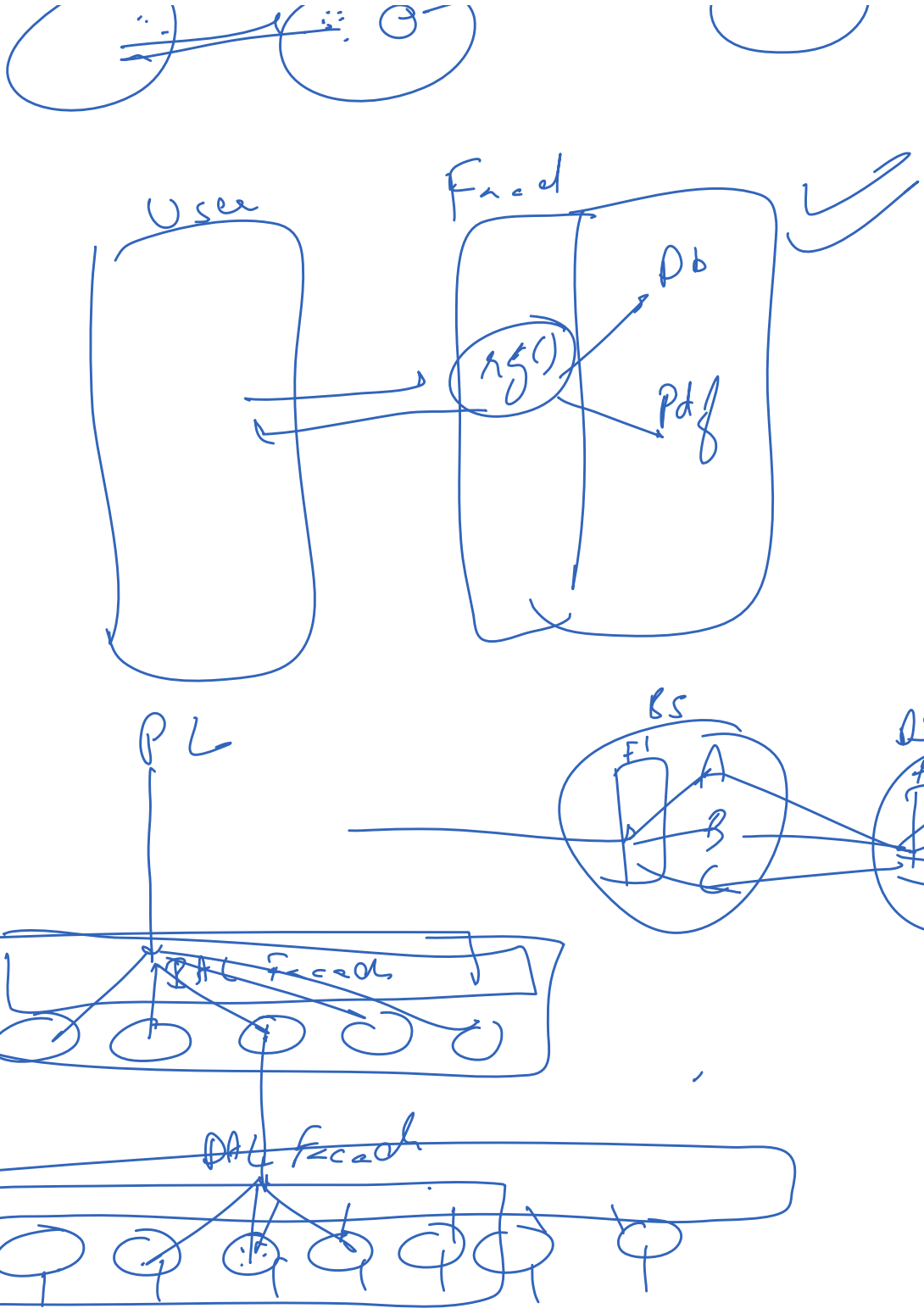





Beginner.
App
DP
Software
IA
A1
A
IB
B
B1
C
C1
Library.
Fac
Idb
Mysql
SQL
Idb db = new
IRS
HTML
Pdf
User.
Fac ch
main()
A
B
C
Solution
Problem


Pkg
Push
Pkg
Feed
A
B
Pkg
End


C1
EU
C1
C2
EU

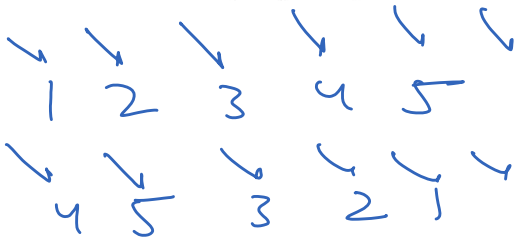


9:12 - 9:22 Try

946. Validate Stack Sequences

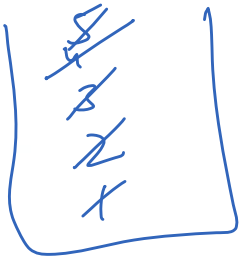
Medium 3755 64 Add to List Share

Given two integer arrays `pushed` and `popped` each with distinct values, return `true` *if this could have been the result of a sequence of push and pop operations on an initially empty stack, or* `false` *otherwise.*

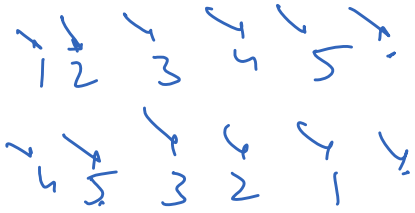


Example 1:

```
Input: pushed = [1,2,3,4,5], popped = [4,5,3,2,1]
Output: true
Explanation: We might do the following sequence:
push(1), push(2), push(3), push(4),
pop() -> 4,
push(5),
pop() -> 5, pop() -> 3, pop() -> 2, pop() -> 1
```



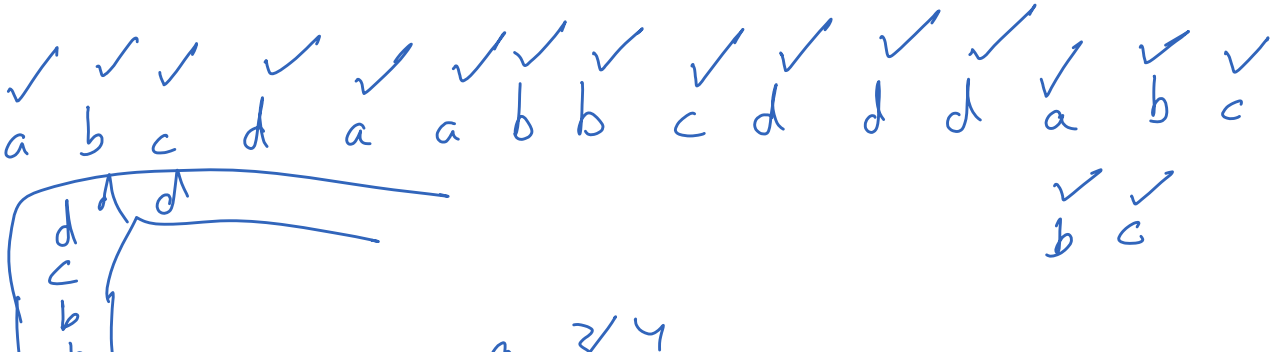
Example 2:

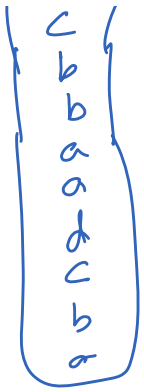


```
while(i < pushed.length){
    st.push(pushed[i]);
    i++;

    while(st.size() > 0 && j < popped.length && st.peek() == popped[j]){
        st.pop();
        j++;
    }
}

return j == popped.length;
```





a - 3 4
b - X X 3 4 5
c - 1 2 3 4
d - X X 3 4

895. Maximum Frequency Stack

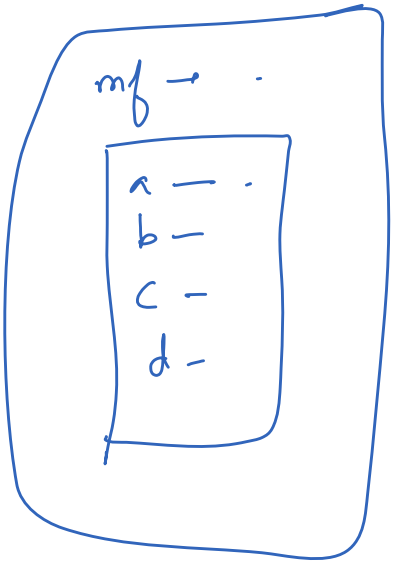
Hard 3927 57 Add to List Share

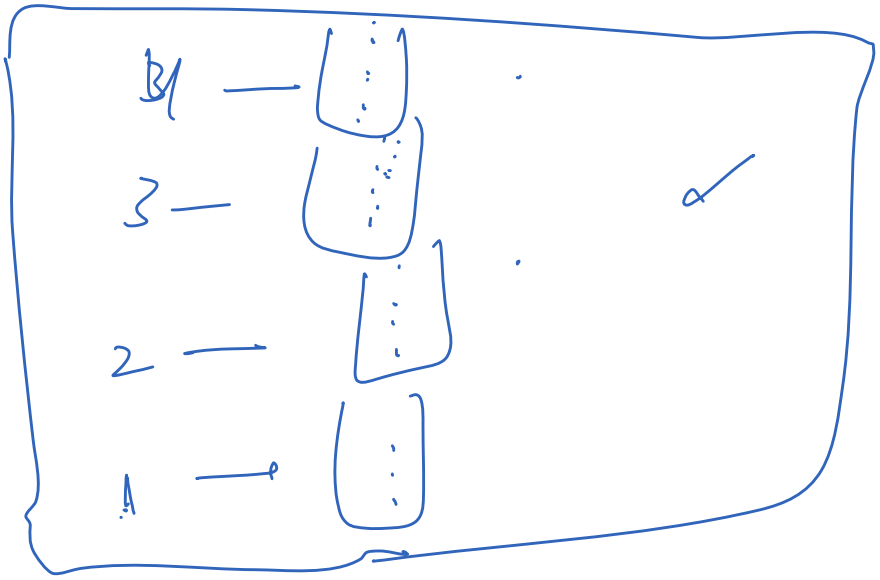
Design a stack-like data structure to push elements to the stack and pop the most frequent element from the stack.

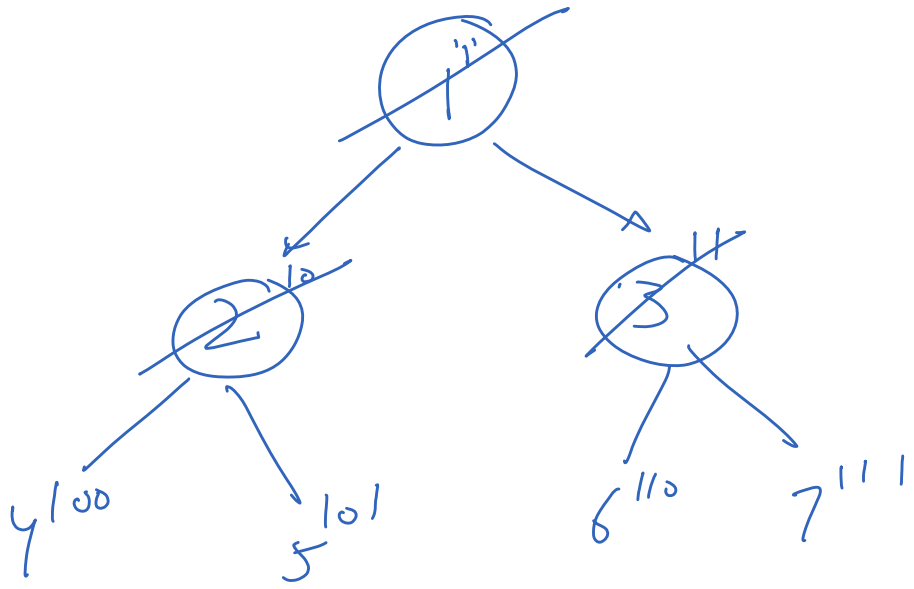
Implement the `FreqStack` class:

- `FreqStack()` constructs an empty frequency stack.
- `void push(int val)` pushes an integer `val` onto the top of the stack.
- `int pop()` removes and returns the most frequent element in the stack.
    - If there is a tie for the most frequent element, the element closest to the stack's top is removed and returned.

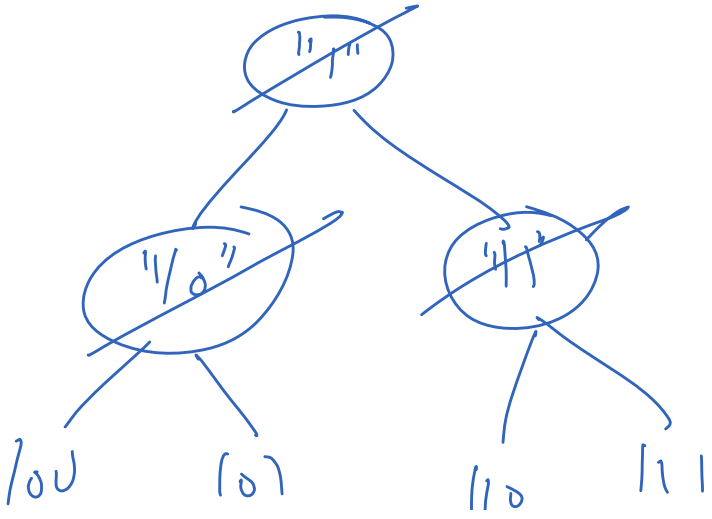
a b a a c c d b d d a b b c







1 → 1
2 → 10
3 → 11



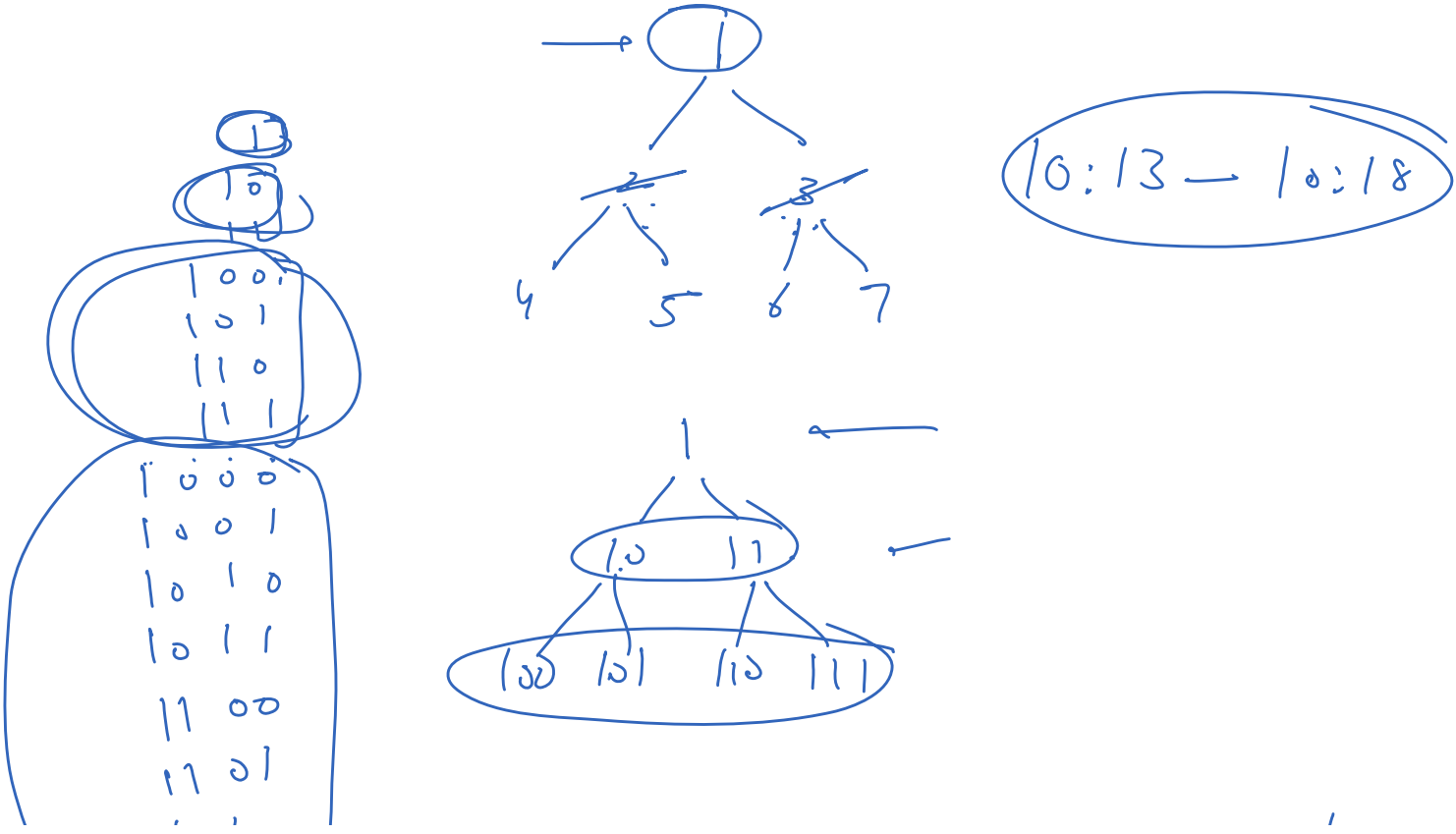
Generate Binary Numbers

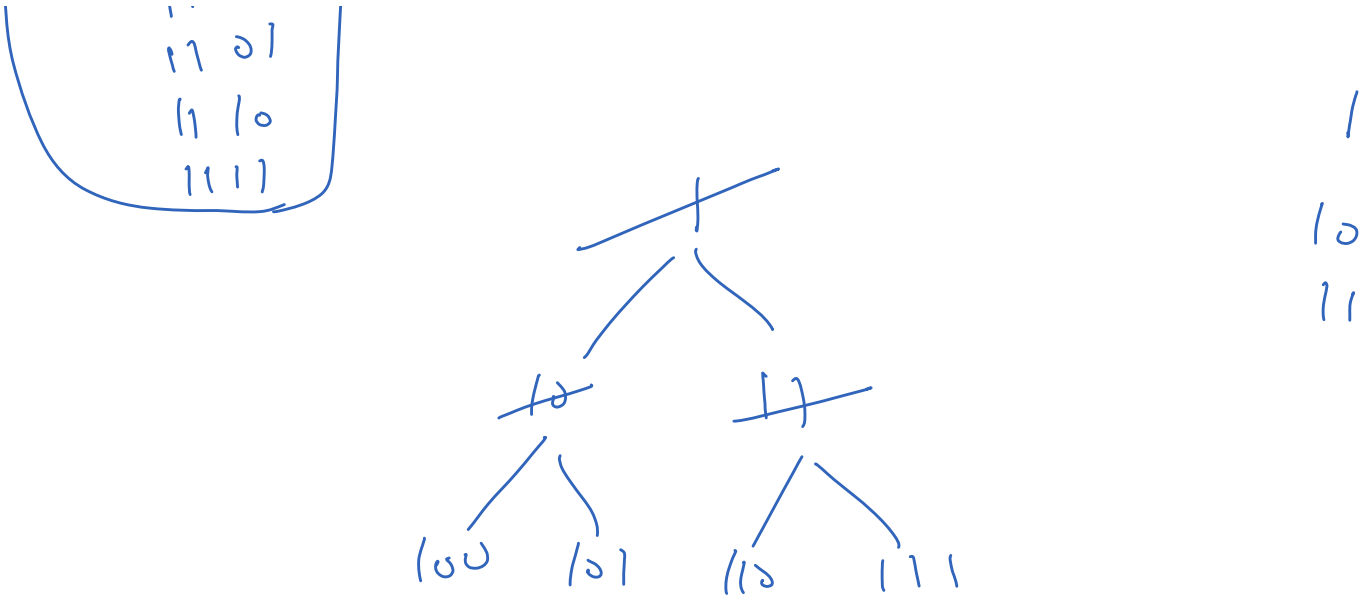
Basic Accuracy: **58.36%** Submissions: **22695** Points: **1**

Given a number **N**. The task is to generate and print all **binary numbers with decimal values** from **1 to N**.

Example 1:

Input:
N = 2
Output:
1 10
Explanation:
Binary numbers from
1 to 2 are 1 and 10.





312. Burst Balloons

Hard 6761 172 Add to List Share

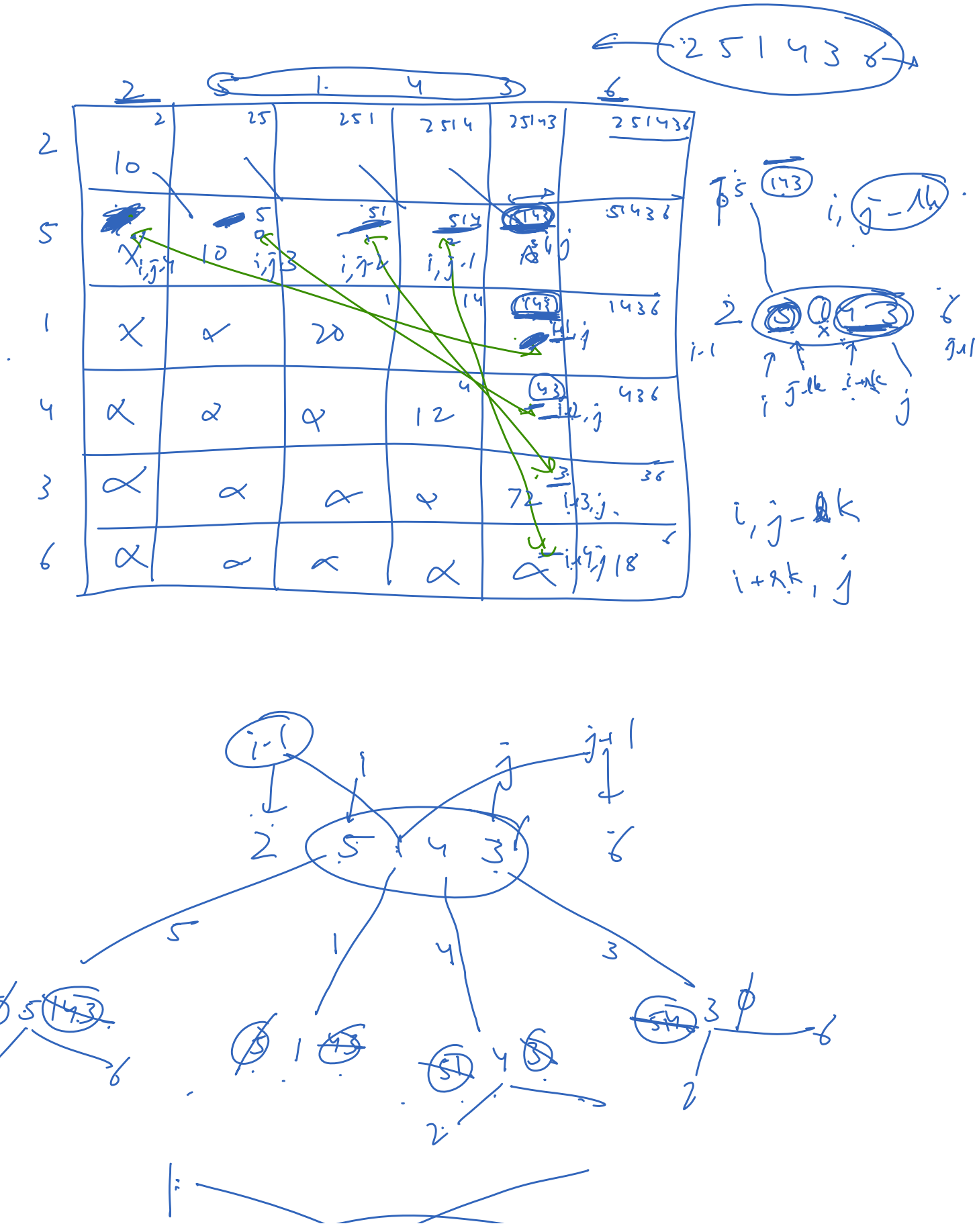
You are given `n` balloons, indexed from `0` to `n - 1`. Each balloon is painted with a number on it represented by an array `nums`. You are asked to burst all the balloons.

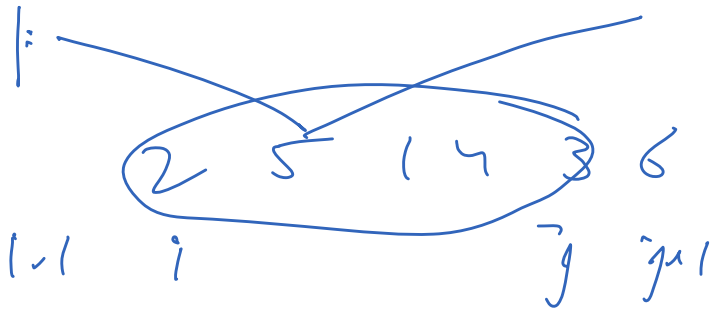
If you burst the i^{th} balloon, you will get `nums[i - 1] * nums[i] * nums[i + 1]` coins. If `i - 1` or `i + 1` goes out of bounds of the array, then treat it as if there is a balloon with a `1` painted on it.

Return *the maximum coins you can collect by bursting the balloons wisely.*

Example 1:

```
Input: nums = [3,1,5,8]
Output: 167
Explanation:
nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []
coins =  3*1*5    +   3*5*8   +  1*3*8  + 1*8*1 = 167
```





1039. Minimum Score Triangulation of Polygon

Medium 1248 125 Add to List Share

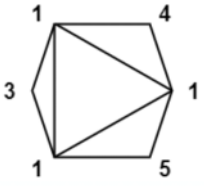
You have a convex `n` -sided polygon where each vertex has an integer value. You are given an integer array `values` where `values[i]` is the value of the i^{th} vertex (i.e., **clockwise order**).

You will **triangulate** the polygon into `n - 2` triangles. For each triangle, the value of that triangle is the product of the values of its vertices, and the total score of the triangulation is the sum of these values over all `n - 2` triangles in the triangulation.

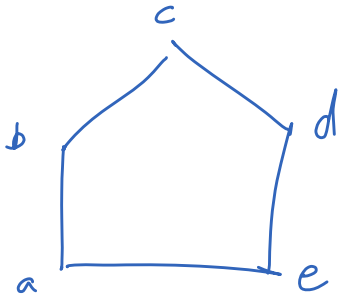
Return *the smallest possible total score that you can achieve with some triangulation of the polygon.*

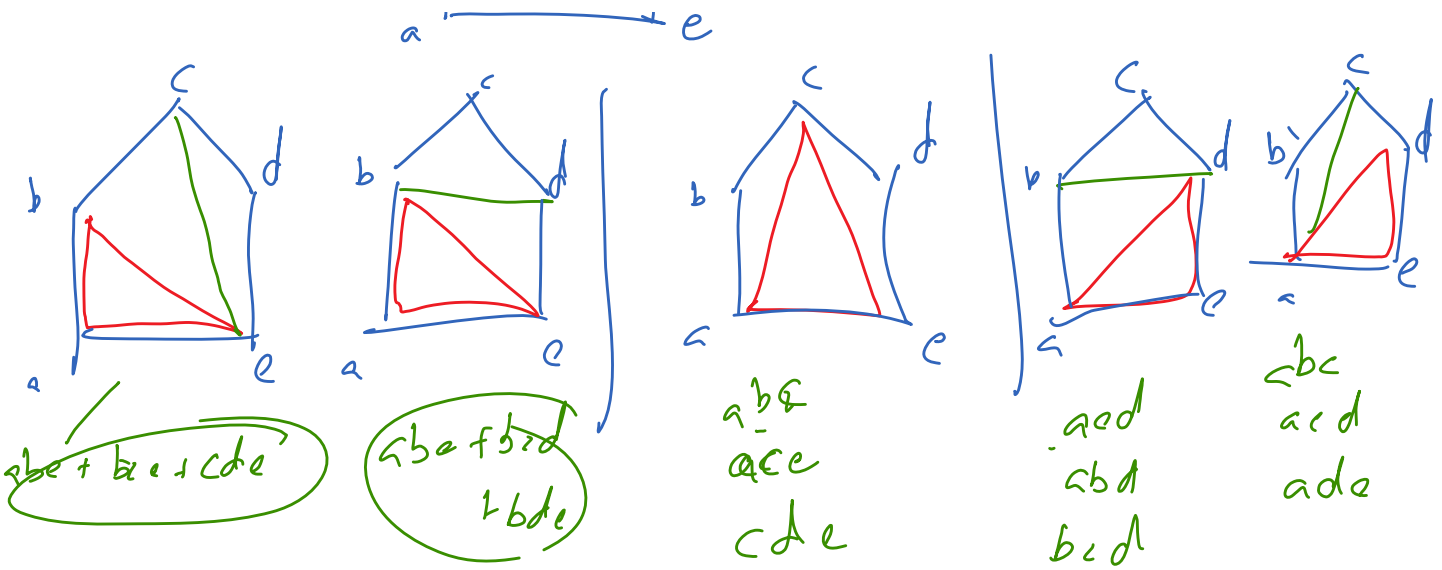
11:16 - 11:26

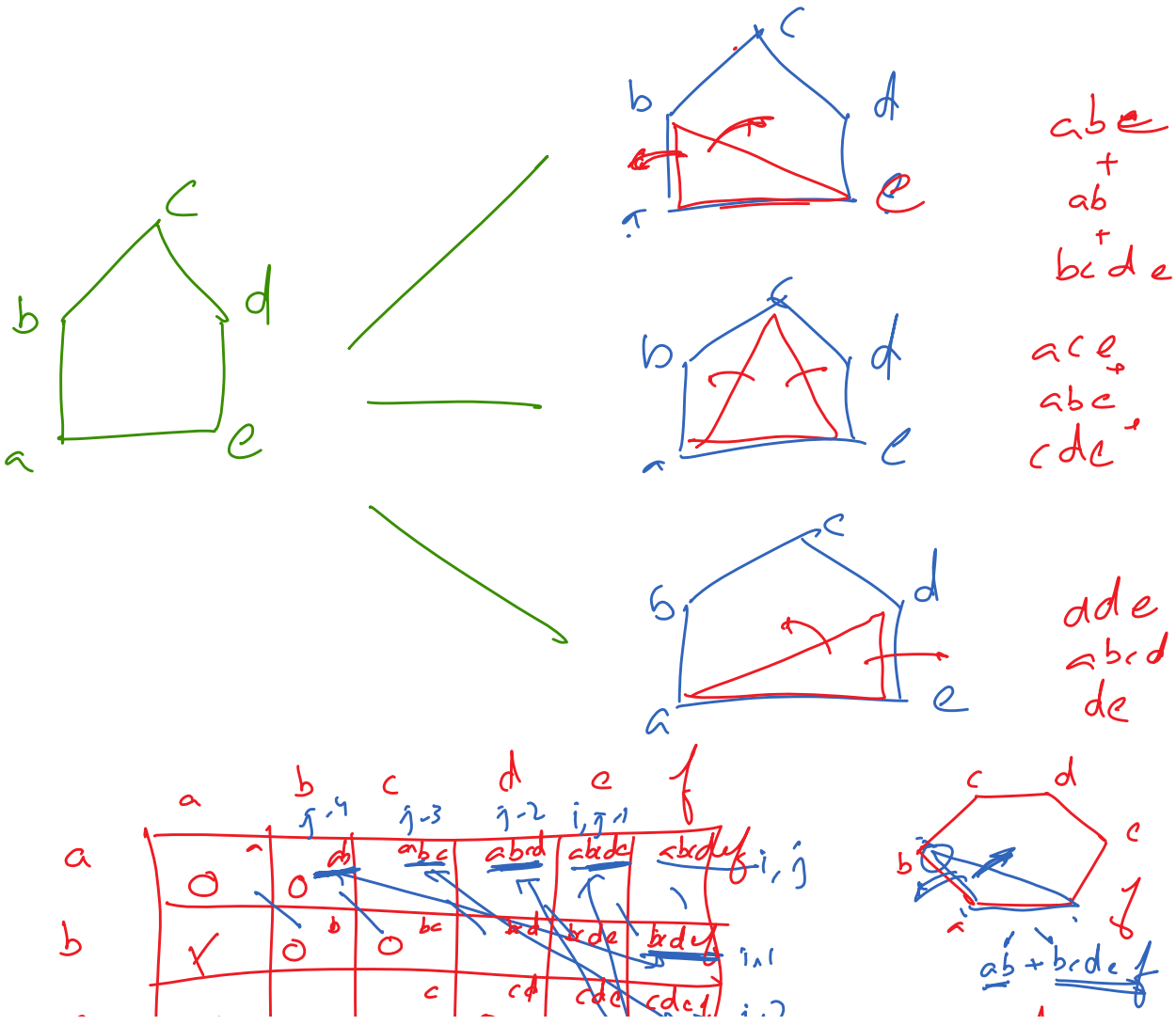
Example 3:




```
Input: values = [1,3,1,4,1,5]
Output: 13
Explanation: The minimum score triangulation has score 1*1*3 + 1*1*4 + 1*1*5 + 1*1*1 =
13.
```




a
e
c
b
d
ab + bc + cde
abc + bcd
bde
abe
ace
cde
aed
abd
bcd
bc
acd
ade


c
b
d
a
e
abe + ab + bcde
ace + abc + cde
dde + abcd + de
i, j
i-1, j
i, j-1
ab + bcde