HEAP ...1

Video-14

(Priority Queue)

codestorywithMIK

(Priority Queue)

codestorywithMIK

Leetcode
-621

Medium

Easy.

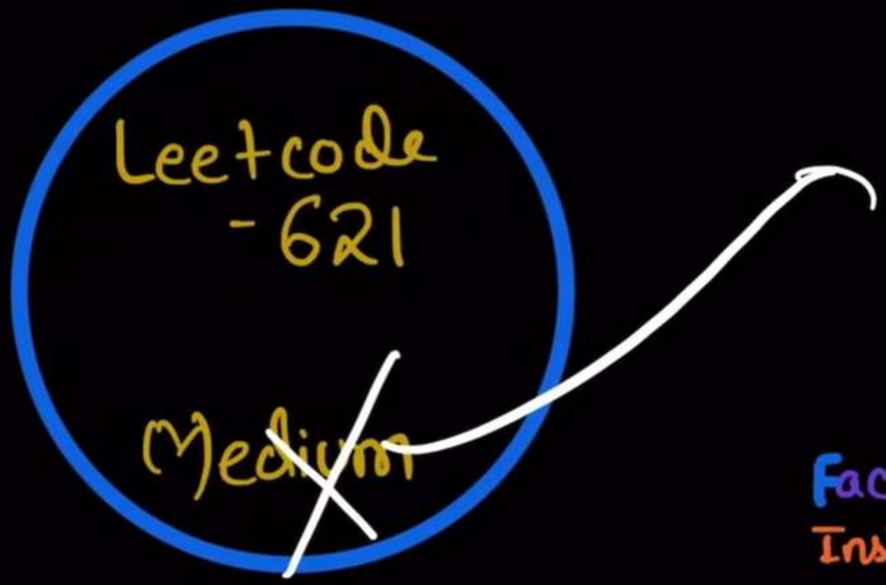
① PQ
② Greedy.

Facebook
Instagram → codestorywithMIK

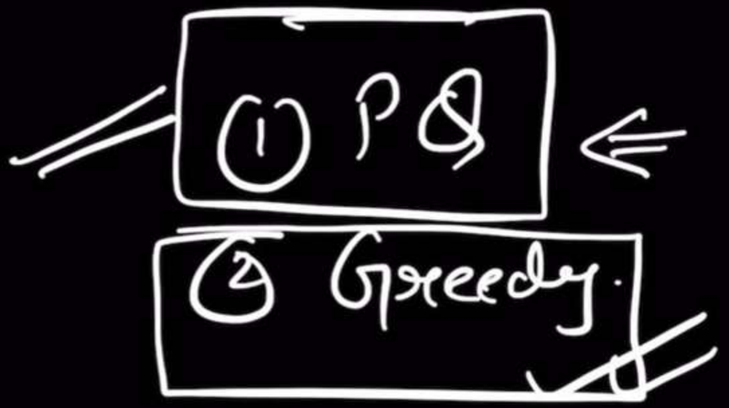
Twitter → cswithMIK

→ codestorywithMIK

codestorywithMIK



easy.



Facebook
Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

Company :- Meta

621. Task Scheduler

Medium | Topics | Companies

You are given an array of CPU `tasks`, each represented by letters A to Z, and a cooling time, `n`. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: **identical** tasks must be separated by at least `n` intervals due to cooling time.

Return the *minimum number of intervals* required to complete all tasks.

621. Task Scheduler

Medium | Topics | Companies

You are given an array of CPU `tasks`, each represented by letters A to Z, and a cooling time, `n`. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: **identical** tasks must be separated by at least `n` intervals due to cooling time.

Return the *minimum number of intervals* required to complete all tasks.

A _ _ 1

Example :- { 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

Output = 8

621. Task Scheduler

Medium | Topics | Companies

You are given an array of CPU `tasks`, each represented by letters A to Z, and a cooling time, `n`. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: **identical** tasks must be separated by at least `n` intervals due to cooling time.

Return the *minimum number of intervals* required to complete all tasks.

A _ _ A _ _ A

Example :- { 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

Output = 8

621. Task Scheduler

Medium | Topics | Companies

You are given an array of CPU `tasks`, each represented by letters A to Z, and a cooling time, `n`. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: **identical** tasks must be separated by at least `n` intervals due to cooling time.

Return the *minimum number of intervals* required to complete all tasks.

A B _ A B _ A

Example :- { 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

Object Eraser

Output = 8

621. Task Scheduler

Medium | Topics | Companies

You are given an array of CPU `tasks`, each represented by letters A to Z, and a cooling time, `n`. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: **identical** tasks must be separated by at least `n` intervals due to cooling time.

Return the *minimum number of intervals* required to complete all tasks.

A B ~~A~~ A B _ A

Example :- { 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

Output = 8

621. Task Scheduler

Medium | Topics | Companies

You are given an array of CPU `tasks`, each represented by letters A to Z, and a cooling time, `n`. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: **identical** tasks must be separated by at least `n` intervals due to cooling time.

Return the *minimum number of intervals* required to complete all tasks.

A B i A B i A B

Example :- { 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

Output = 8

Example :- { 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

Output = 8

{ 'A', 'C', 'A', 'B', 'D', 'B' }, n = 1

Output = 6

A B A B C D

Thought Process :-

{ 'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A' }, n=1

Merge Process

{ 'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A' }, n=1

A = 6

B = 2

C = 1

D = 1

Hough Process

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A'}, n=1

C D

A = 6

B = 2

~~C = 1~~

~~D = 1~~

Hoare Process

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B

A = 6

B = 2

~~C = 1~~

~~D = 1~~

Mongo DB Process

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A

A = 6

B = 2

~~C = 1~~

~~D = 1~~

Pen

Thought Process:

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _

A = 6

B = 2

~~C = 1~~

~~D = 1~~

Merge Process

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ _ A _ _ A _ _ A . . .

A = 6

B = 2

~~C = 1~~

~~D = 1~~

Object Eraser

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _ A

A = 6

B = 2

~~C = 1~~

~~D = 1~~

Object Eraser

1 U

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

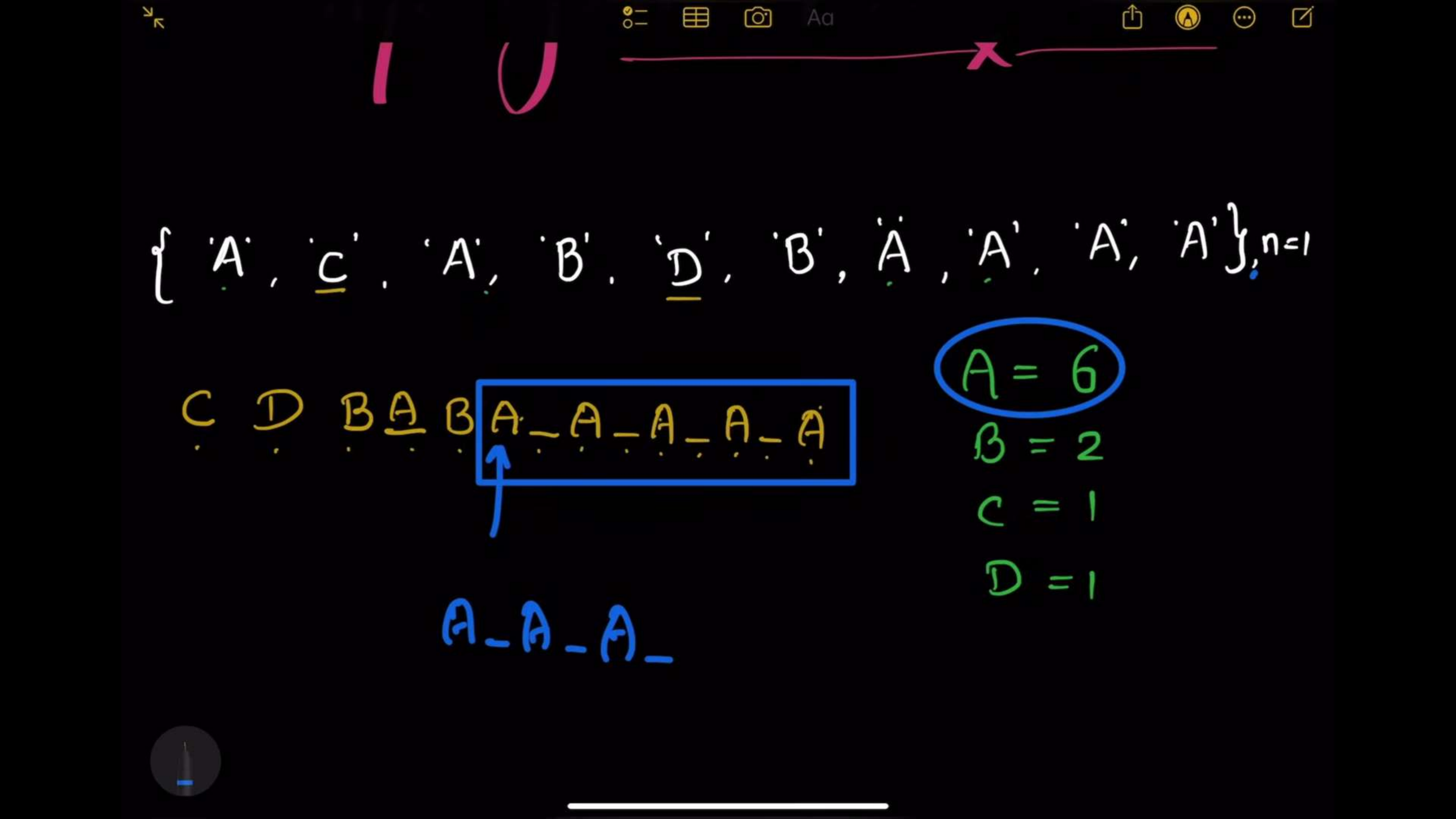
C D B A B A _ A _ A _ A _ A

A = 6

B = 2

C = 1

D = 1


{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1
C D B A B A _ A _ A _ A _ A
A = 6
B = 2
C = 1
D = 1
A _ A _ A _

10

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _ A

A B

A = ~~6~~ 5

B = ~~2~~ 1

C = 1

D = 1

1 0

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _ A

A B A

A = ~~6~~ ~~8~~ 4

B = ~~2~~ 1

C = 1

D = 1

10

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _ A

A B A B

A = ~~6~~ ~~8~~ 4

B = ~~2~~ 1 0

C = 1

D = 1

1 0

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _ A

A B A B A C ∧

A = ~~6~~ ~~8~~ 4

B = ~~2~~ ~~1~~ 0

C = ~~1~~ 0

D = 1

10

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _ A

A B A B A C A D

A = ~~6~~ ~~8~~ 4

B = ~~2~~ ~~1~~ 0

C = ~~1~~ 0

D = ~~1~~ 0

1 0

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A _ A

A B A B A C A D A _ A

A = ~~6~~ ~~8~~ ~~4~~

B = ~~2~~ ~~1~~ 0

C = ~~1~~ 0

D = ~~1~~ 0

I U

{'A', 'C', 'A', 'B', 'D', 'B', 'A', 'A', 'A', 'A'}, n=1

C D B A B A _ A _ A _ A / A

A B A B A C A D A _ A

A = 6 8 4 0
B = 2 1 0
C = 1 0
D = 1 0

Object Eraser

C D B B B A _ A _ A _ A _ A

A B A B A C A D A _ A

B = 2 X 0

C = X 0

D = X 0

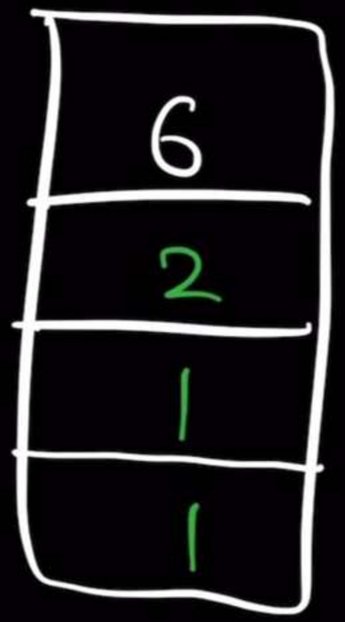
Greedy + P.Q.
(map).

A = 6

B = 2

C = 1

D = 1



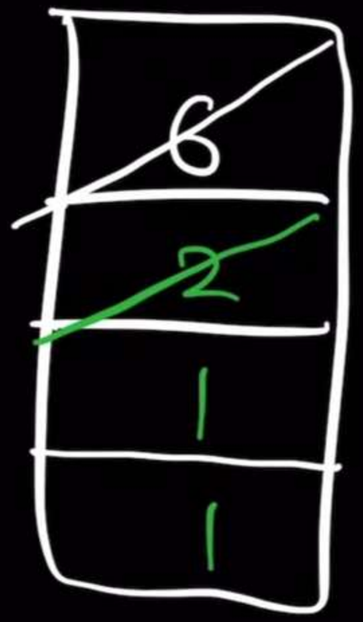
, n = 1

A = 6

B = 2

C = 1

D = 1



n = 1

freq = ~~6~~ 5

freq = 2 1

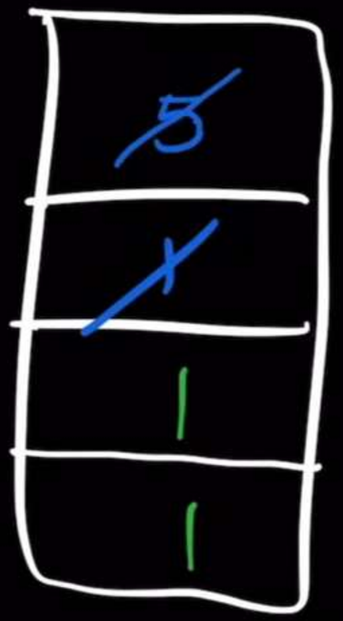
A B

A = 6

B = 2

C = 1

D = 1



n = 1

freq = 5 4→

freq = 1 0→ {4, 0}

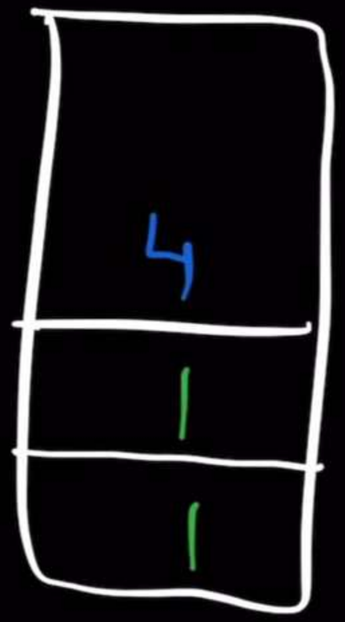
A B | A B |

A = 6

B = 2

C = 1

D = 1



n = 1

freq = 4

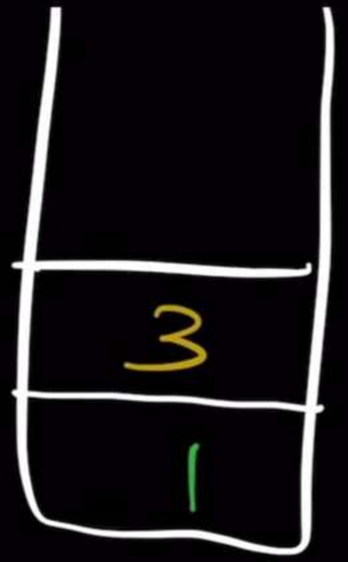
freq =

A B | A B |

B = 2

C = 1

D = 1



n = 1

freq = 3

freq = 0

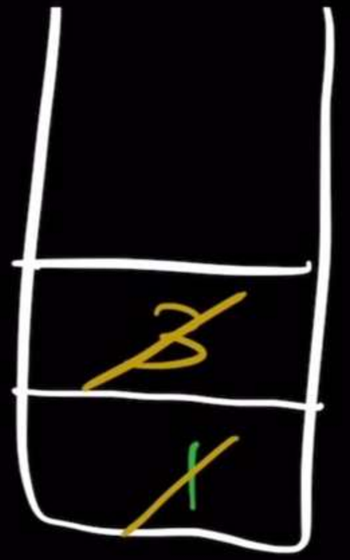
A B | A B | A C

2 2 2

B = 2

C = 1

D = 1



n = 1

freq = ~~B~~ 2

freq = ~~1~~ 0

{2, 0}

A B | A B | A C | A D

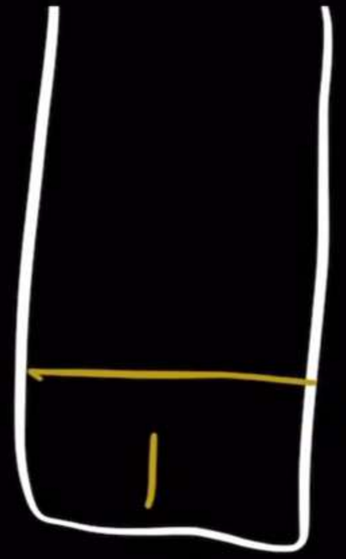
2 2 2

Object Eraser

B = 2

C = 1

D = 1



n = 1

freq = ~~2~~ 1

freq =

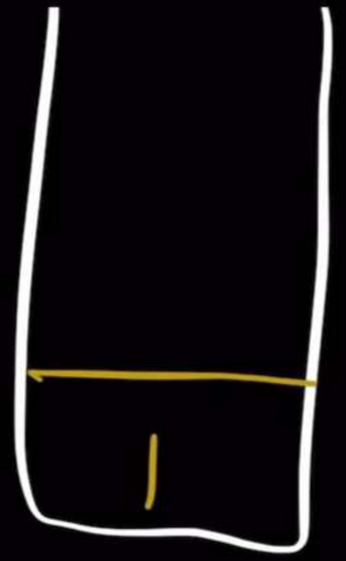
A B | A B | A C | A D | A

2 2 2 2

B = 2

C = 1

D = 1



n = 1

freq = ~~2~~ 1

freq =

A B | A B | A C | A D | A ? |

→ 2 2 2 2 2

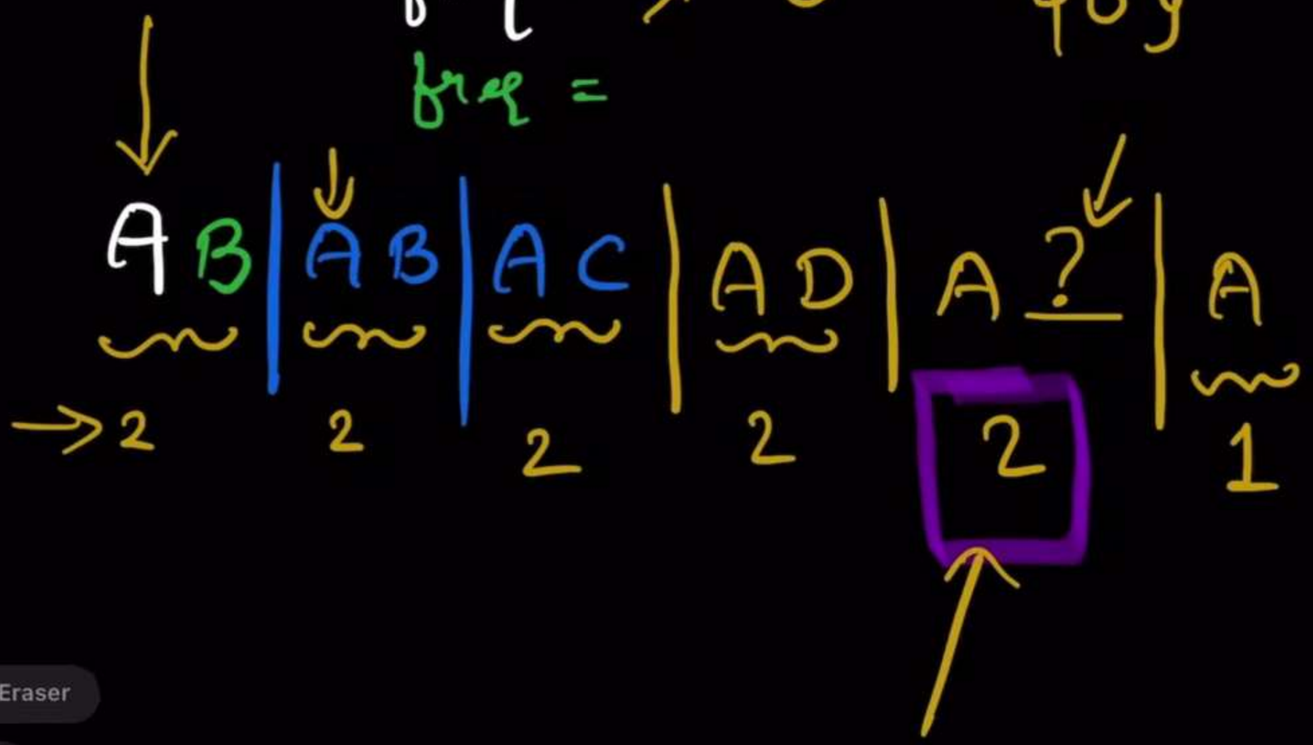
C = 1

D = 1

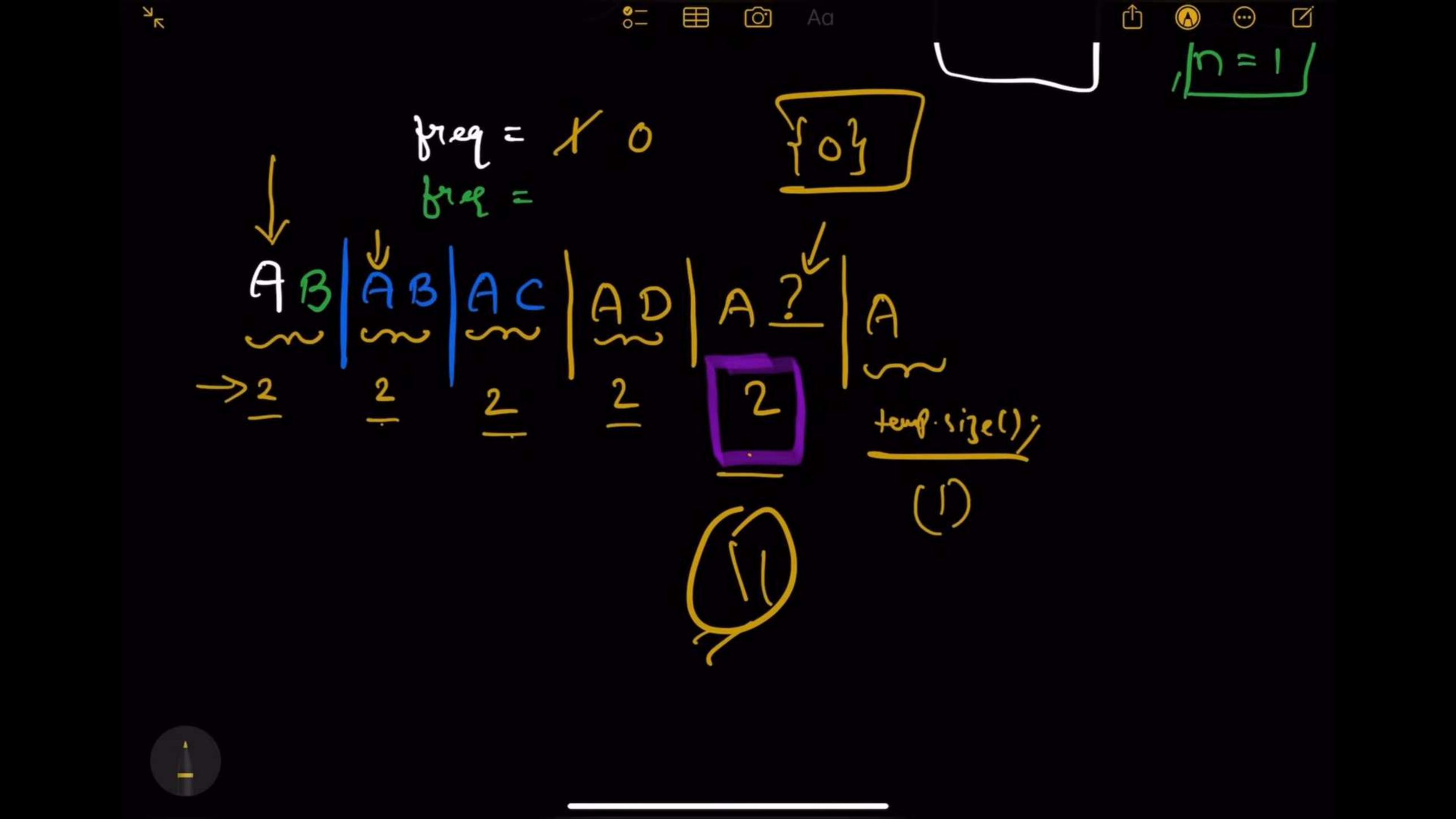
n = 1

freq = X 0 {0}

freq =



Object Eraser

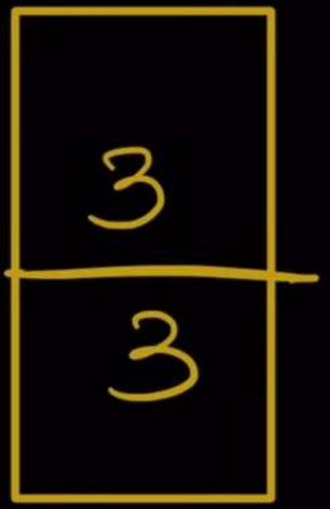

freq = X 0
freq =
{0}
n = 1
A B | A B | A C | A D | A ? | A
2 2 2 2 2
temp.size();
(1)

{'A', 'A', 'A', 'B', 'B', 'B'}, n=2

A = 3

B = 3

{ 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2



A = 3

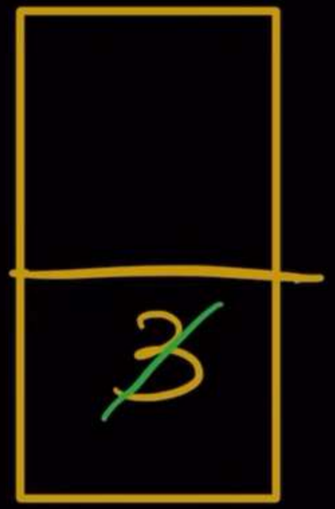
B = 3

, n = 2

{ 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

freq = ~~3~~ 2

freq = ~~3~~ 2 {2



A = 3

B = 3

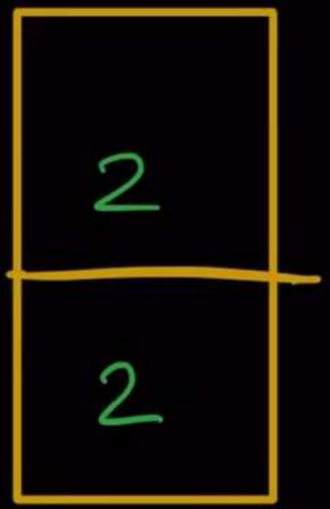
, n = 2

AB

freq = 2

freq = 2

{2, 2}



A = 3

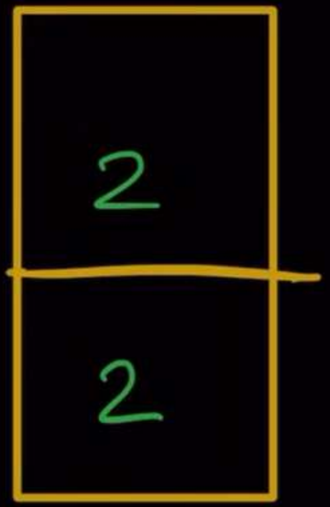
B = 3

, n = 2

AB_

freq = 2

freq = 2 {2, 2}



A = 3

B = 3

, n = 2

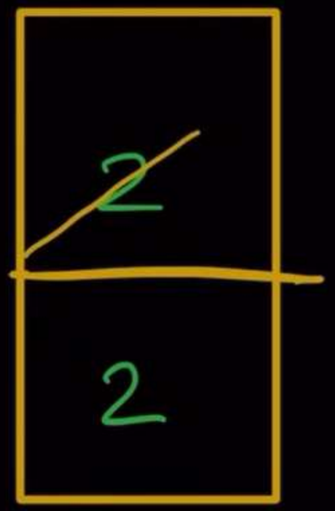
AB_

n+1

[A, A, A, B, B, C], n = 6

freq = ~~2~~ 1

freq =



A = 3

B = 3

, n = 2

AB_ | n

n+1

[A, A, A, B, B, C], n = 6

freq = ~~2~~ 1

freq = ~~2~~ 1

A = 3

B = 3

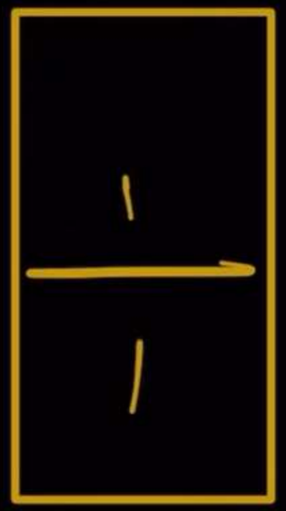
, n = 2

AB_ | AB_

n+1

freq = ~~2~~ 1

freq = ~~2~~ 1



A = 3

B = 3

, n = 2

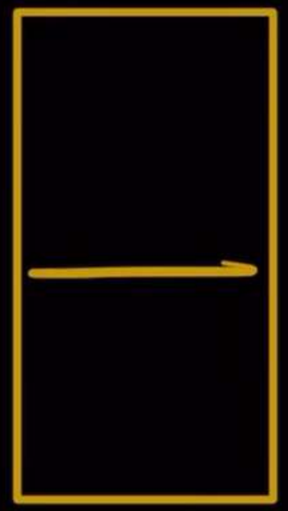
AB_ | AB_

n+1

Object Eraser

freq = X 0

freq = X 0 {0,0}



A = 3

B = 3

, n = 2

$\underbrace{AB\_}_{n+1} \mid \underbrace{AB}_{n+1}\_ \mid AB\_$

freq = X 0

freq = X 0 {0,0}

A = 3

B = 3

, n = 2

AB_ | AB_ | AB

n+1 | n+1 | tempSize()

if (Pimply)

freq = X 0

freq = X 0 {0,0}

A = 3

B = 3

, n = 2

AB_ | AB_ | AB⊖

n+1 | n+1 | tempSize()

if (Pimply)

Pen

freq = X 0

freq = X 0 {0,0}

A = 3

B = 3

, n = 2

AB_ | AB_ | AB

n+1 n+1

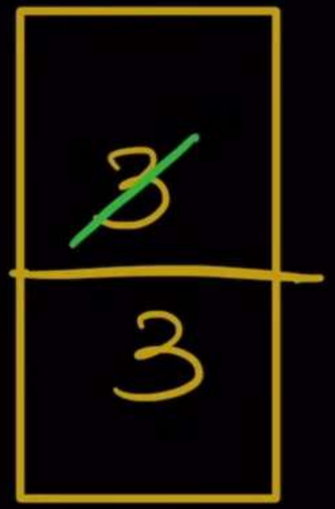
tempSize()

if (Pushing)

{ 'A', 'A', 'A', 'B', 'B', 'B' }, n = 2

freq = ~~3~~ 2

f2



A = 3

B = 3

, n = 2

A