PROBLEM-I : ①

Given classes: Side, A, B, C, D

As we have 4 classes A, B, C, D So the no of bindings for polymorphic testing calls are 4 Bindings.

Hence.

Object Pa Bindings	pb Bindings	pc Bindings	pd Bindings
* object of A	* object of A	* object of A	* object of A
* Object of B	* object of B	* object of B	* object of B
* object of C	* object of C	* object of C	* object of C
* object of D	* object of D	* object of D	* object of D.

Polymorphic Testing at Statement 10: If (pa→get() > pb→get()){

Pa:

object of A: Test#1: 12, 13, 14, 0

object of B: Not Possible

object of C: Not Possible

object of D: Not Possible

Pb:

object of A: Not Possible

object of B: Test#2: 12, 13, 14, 0

object of C: Not Possible

object of D: Not Possible.

Polymorphic Testing at Statement 14: If (pa→get() > pc→get())

Pa:

object of A: Test#3: 12, 13, 14, 0

object of B: Test#4: 13, 12, 14, 0

object of C: Not Possible

object of D: Not Possible

Pc

object of A: Not Possible

object of B: Not Possible

object of C: Test#5: 12, 13, 14, 0

object of D: Test#6: 12, 13, 14, 0

Polymorphic Testing at Statement 18: if(pb→get() > pc→get()){

pb

Object of A: Test #7: 13, 12, 14, 0
Object of B: Test #8: 12, 13, 14, 0
object of C: Not Possible
object of D: Not Possible

PC

Object of A: Test #9: 12, 13, 11, 1
Object of B: Test #24: 13, 12, 11, 1
Object of C: Test #10: 12, 13, 14, 1
Object of D: Test #11: 12, 13, 14, 0.

Polymorphic Testing at Statement 22: if(pa→get() + pb→get() <= pc→get())

Pa

Object of A: Test #12: 12, 13, 14, 0
Object of B: Test #13: 13, 12, 14, 0
Object of C: Test #14: 12, 13, 11, 1
Object of D: Test #15: 12, 13, 11, 0

pb

Object of A: Test #16: 13, 12, 14, 0
Object of B: Test #17: 12, 13, 14, 0
Object of C: Test #18: 12, 14, 13, 1
Object of D: Test #19: 12, 14, 13, 0

PC

Object of A: Test #20: 12, 12, 11, 1
Object of B: Test #21: 12, 14, 13, 1
Object of C: Test #22: 12, 13, 14, 1
Object of D: Test #23: 12, 13, 14, 0.

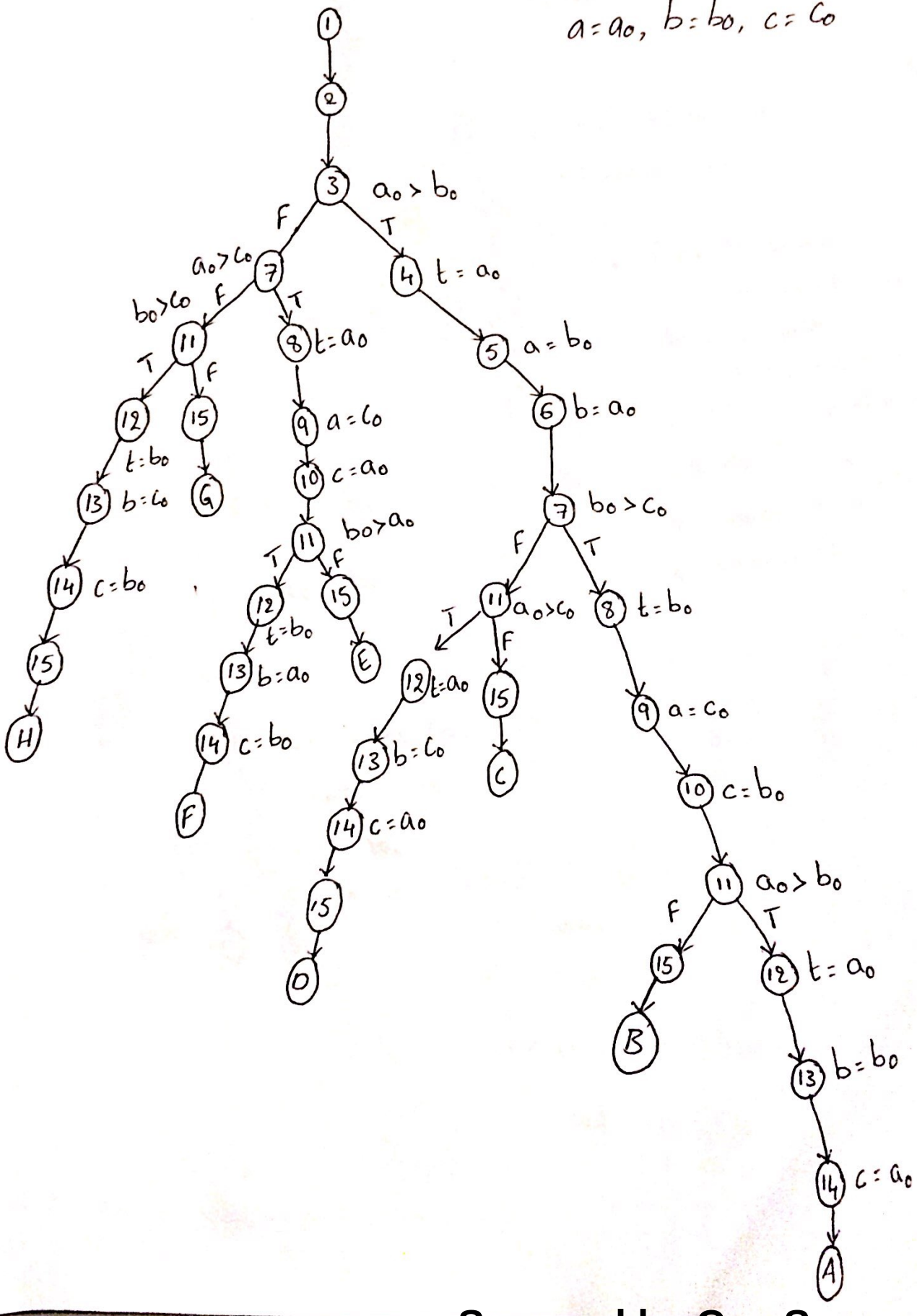
Problem - II

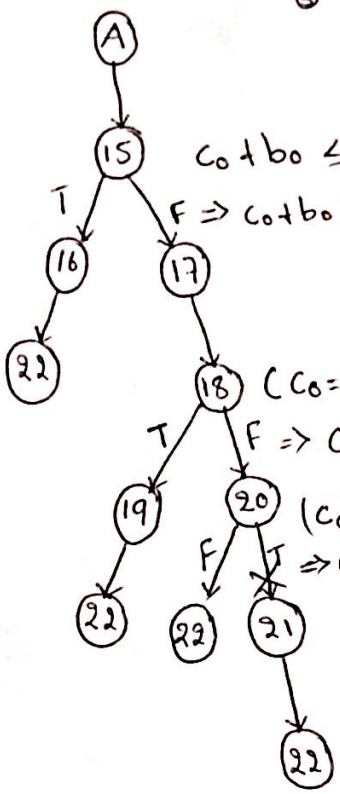
(2)

Symbolic - Execution Tree

Consider

$a = a_0,\ b = b_0,\ c = c_0$





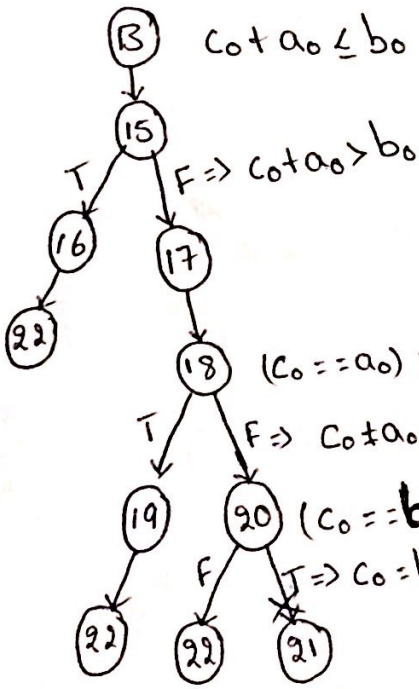
From (7) $b_0 > c_0$

(11) $a_0 > b_0$ as both are in True path.

$\Rightarrow a_0 > b_0 > c_0$

Which is contradicting with (20) $a_0 = c_0$

Hence path (20) → (21) is not executable

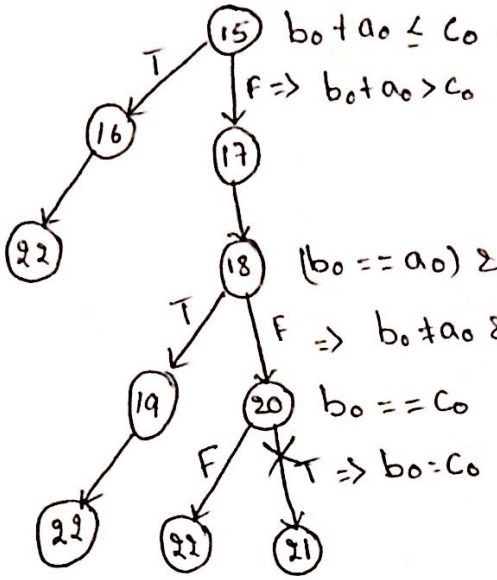


From (7) $b_0 > c_0$ T is contradicting With

(20) T $b_0 = c_0$

Hence path (20) → (21) is not executable.

~~(B)~~ (C)



From Statement

(11) F => $a_0 <= c_0$

(7) F => $b_0 <= c_0$

(3) T => $a_0 > b_0$

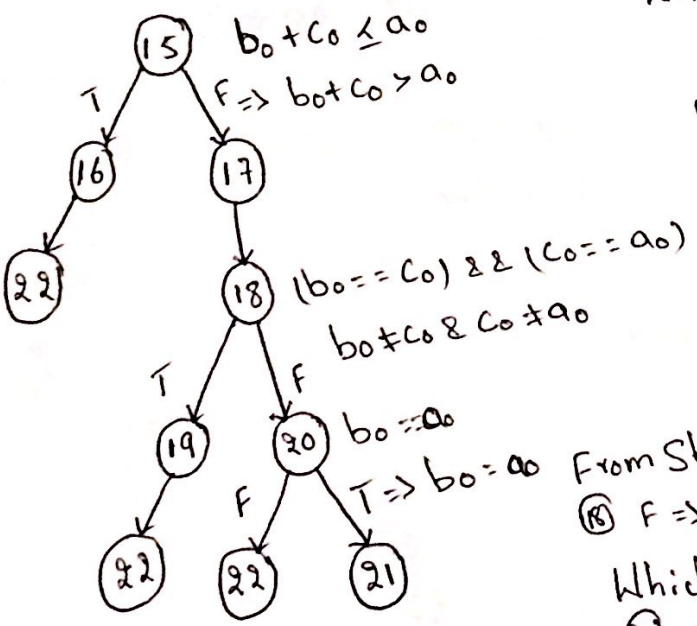
(18) F => $a_0 \neq c_0$

Consolidating all these conditions

We get. $b_0 < a_0 < c_0$

Which is contradicting with (20) T $b_0 = c_0$. Hence path (20) → (21) is not executable.

(D)



From Statement

(18) F => $c_0 \neq a_0$ & $b_0 \neq c_0$, (11) T => $a_0 > c_0$

(7) F => $b_0 <= c_0$

Which is contradicting with (3) $a_0 > b_0$ (T)

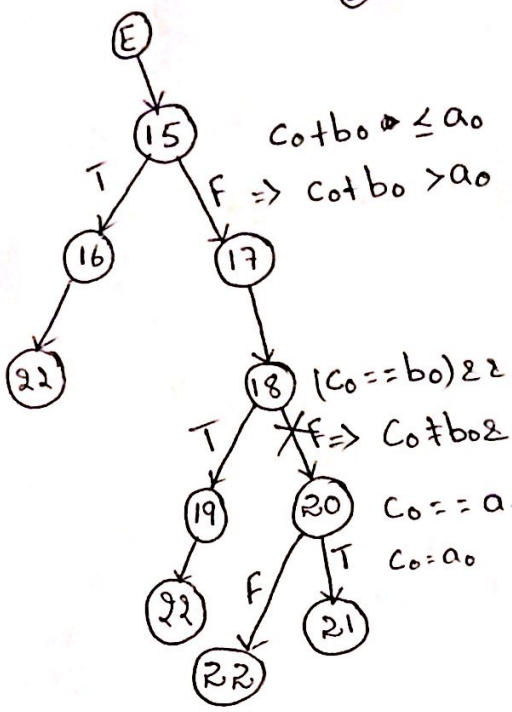
(20) T for consolidated value

$b_0 < a_0 < c_0$

with $b_0 = a_0$,

Hence path (20) → (21) is not executable.

③



From Statements:

(11) F $b_0 <= a_0$

(7) T $a_0 > c_0$

(3) F $a_0 <= b_0$

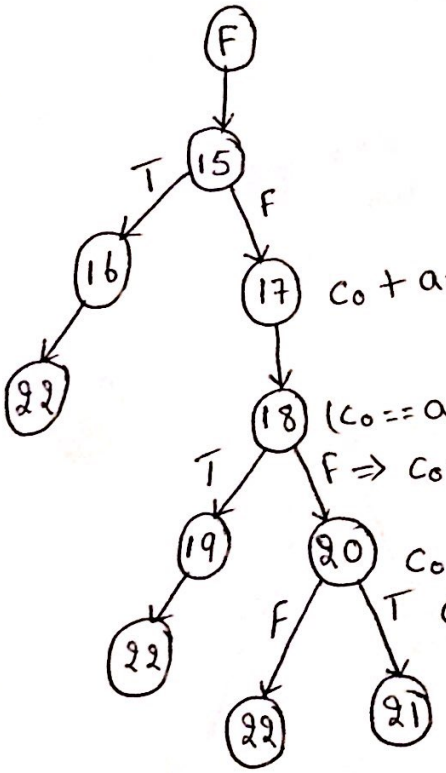
(18) F $c_0 \neq b_0$ & $b_0 \neq a_0$

By Consolidating all these We get Contradiction for (11), (3) and (18) Statements

i.e; From (11) & (3) we can infer $a_0 = b_0$

But (18) Contradicts saying $a_0 \neq b_0$

Hence this path is not executed



From Statements

(11) T $b_0 > a_0$

(7) T $a_0 > c_0$

(3) F $a_0 <= b_0$

(18) F $c_0 \neq a_0$ & $a_0 \neq b_0$

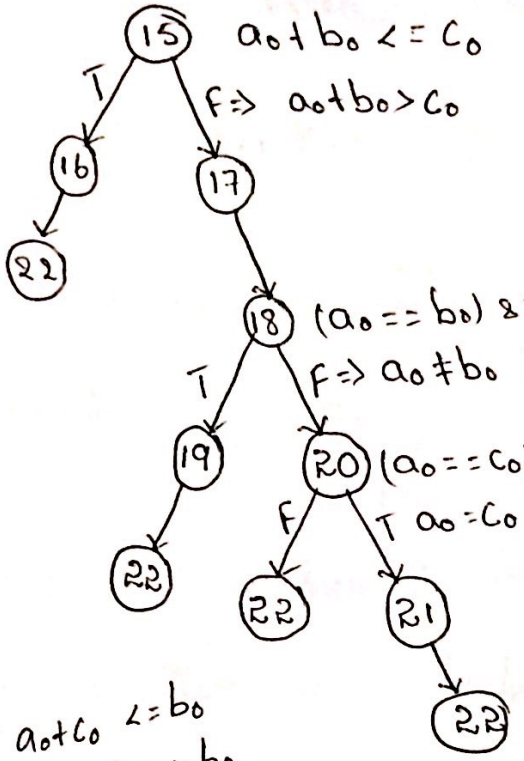
By Consolidating We get

$c_0 < a_0 < b_0$

Whic contradicts Statement (20) T $c_0 = b_0$

Hence this path is not executed.

(G)



From Statements

(11) F $b_0 <= c_0$

(7) F $a_0 <= c_0$

(3) F $a_0 <= b_0$

(18) F $a_0 \neq b_0$ or $b_0 \neq c_0$

By Consolidating these conditio

We get

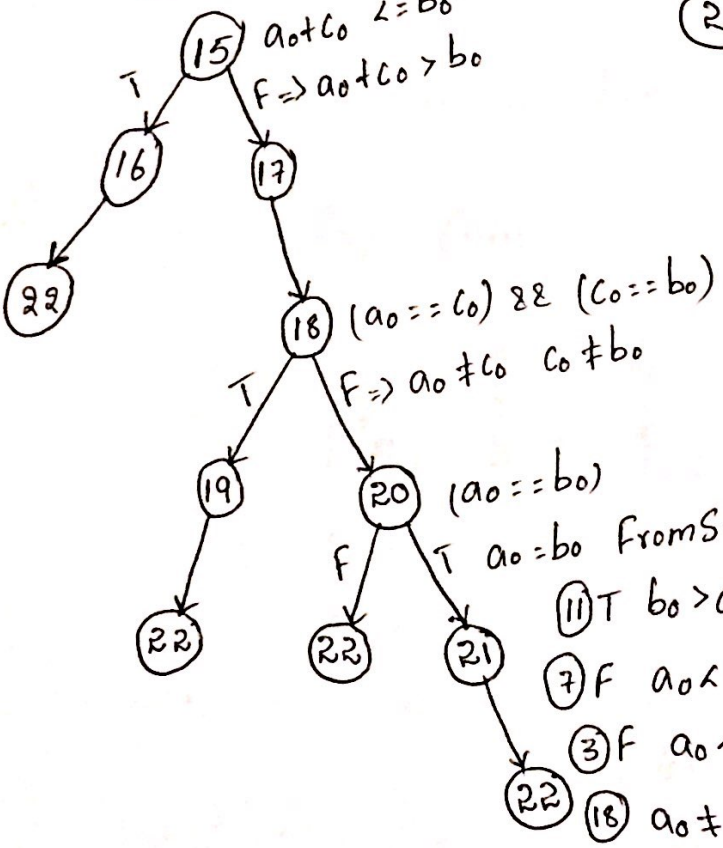
$a_0 < b_0 < c_0$

Which contradicts with (20)

$a_0 = c_0$ Hence Path

(20) → (21) is not executable.

(H)



From Statements

(11) T $b_0 > c_0$

(7) F $a_0 <= c_0$

(3) F $a_0 <= b_0$

(18) $a_0 \neq c_0$ or $c_0 \neq b_0$

By Consolidating these Conditions

We get.

$a_0 < c_0 < b_0$ Which Contradicts with (20) $a_0 = b_0$, Hence path (20) → (21) is not

Problem - III

Given Pre-Condition $1 \le n \le 100$

Post-Condition $S = \sum_{j=1}^{n} (|a[j]| == x)$

Loop Invariant at statement 4 is

Consider $S = \sum_{j=i+1}^{n} (|a[j]| == x)$

Prove the correctness of loop Invariant !!!

Using mathematical induction

K: # of times execution reaches point 4.

① K: 1 loop Entry

Path: 1, 2, 3, 4

$i = n$ $S = 0$ $i > 0$

$\Rightarrow i = n$

$S = 0$

Loop Invariant: $S = \sum_{j=i+1}^{n} (|a[j]| == x)$

Substituting $i = n$ $\Rightarrow S = \sum_{j=n+1}^{n} (|a[j]| == x)$ $\rightarrow S = 0$ at Step 3

No Summation performed Hence $S = 0$ is true //

II

Assume that $S_k = \sum_{j=i_k+1}^{n} (|a[j]| == x)$ is true for some k^{th} Iteration

On Loop Iteration:

We have to show the loop Invariant holds true for $(k+1)^{th}$ Iteration.

Variables that are modified during Iteration is 'j' & 'S'

$$\Rightarrow S_{k+1} = \sum_{j=i_{k+1}+1}^{n} (|a[j]| == x)$$

To prove this we have to consider the four possible paths available during the Iteration.

Path #1: 4, 5, 6, 8, 9, 10, 4

Path #2: 4, 5, 7, 8, 9, 10, 4

Path #3: 4, 5, 6, 8, 10, 4

Path #4: 4, 5, 7, 8, 10, 4.

Case 1: $a[i_k] < 0$ (9) $(-a[i_k] == x)$ $i_{k+1} = i_k - 1$

Path: 4, 5, 6, 8, 9, 10, 4

i_k, S_k $\quad$ $S_{k+1} = S_k + 1$ $\quad$ i_{k+1}, S_{k+1}

Variables:

$$S_{k+1} = S_k + 1$$

$$i_{k+1} = i_k - 1$$

Conditions:

$(a[i_k] < 0) \Rightarrow$ True

$(-a[i_k] == x) \Rightarrow$ True

$$\Rightarrow S_{k+1} = \sum_{j=i_{k+1}+1}^{n} (|a[j]| == x)$$

$$S_k + 1 = \sum_{j=i_k-1+1}^{n} (|a[j]| == x)$$

$$S_k = \sum_{j=i_k}^{n} (|a[j]| == x) - 1$$

$$= \sum_{j=i_k}^{n} (|a[j]| == x) - \sum_{j=i_k}^{i_k} (a[j] == x)$$

Since Summation for i_k is subtracted ⇓ This is because we can re-orded the Summation to $j = i_k$

This can be regrouped to $(a[i_k] == x)$ is true.

$$S_k = \sum_{j=i_{k+1}}^{n} (|a[j]| == x)$$ is true based on the assumption in II

Case 2: Path: 4 → 5 → 7 → 8 → ⑩ (a[i_k] == x) → 9 → 10 → 4

4: i_k, S_k

5: a[i_k] < 0

9: $S_{k+1} = S_k + 1$

10: $i_{k+1} = i_k - 1$

4: i_{k+1}, S_{k+1}

Variables:

$$S_{k+1} = S_k + 1$$
$$i_{k+1} = i_k - 1$$

Conditions:

a[i_k] < 0 => False => a[i_k] >= 0

a[i_k] == x => True.

$$\Rightarrow S_{k+1} = \sum_{j=i_{k+1}+1}^{n} (|a[j]| == x)$$

$$S_k + 1 = \sum_{j=i_k-1+1}^{n} (|a[j]| == x)$$

$$S_k = \sum_{j=i_k}^{n} (|a[j]| == x) - 1$$

$$= \sum_{j=i_k}^{n} (|a[j]| == x) - \sum_{j=i_k}^{i_k} (a[j] == x)$$

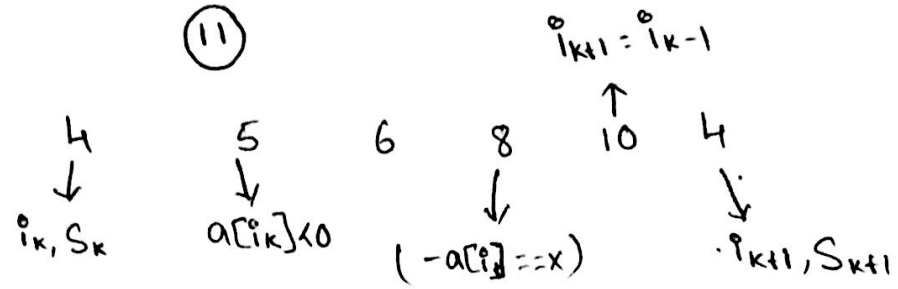
This is because (a[i_k] == x) is true.

Since Summation for i_k is subtracted we can re-ordered the Summation to $j = i_{k+1}$

This can be regrouped to

$S_k = \sum_{j=i_{k+1}}^{n} (|a[j]| == x)$ is true based on the assumption in II

Case 3: Path: 4 → 5 → 6 → 8 → 10 → 4



Variables:

$S_{k+1} = S_k$

$i_{k+1} = i_k - 1$

Conditions:

$a[i_k] < 0 \Rightarrow$ True

$-a[i_k] == x \Rightarrow$ False

$$\Rightarrow S_{k+1} = \sum_{j=i_{k+1}+1}^{n} (|a[j]| == x)$$

$$S_k = \sum_{j=i_{k+1}-1}^{n} (|a[j]| == x)$$

$$S_k = \sum_{j=i_{k+1}}^{n} (|a[j]| == x) + (a[i_k] == x)$$

$\Downarrow$

Since $a[i_k] \neq x$ from Conditions this returns zero

$$S_k = \sum_{j=i_{k+1}}^{n} (|a[j]| == x)$$

is true based on the assumption in II.

Case 4: Path: 4, 5, 7, 8, 10, 4

i_k, S_k ; $a[i_k] < 0$; $(*a[i_k] == x)$; $i_{k+1} = i_k - 1$; i_{k+1}, S_{k+1}

Variables:

$S_{k+1} = S_k$.

$i_{k+1} = i_k - 1$

Conditions:

$a[i_k] < 0 \Rightarrow$ false

$a[i_k] == x \Rightarrow$ false

$$\Rightarrow S_{k+1} = \sum_{j=i_{k+1}+1}^{n} (|a[j]| == x)$$

$$S_k = \sum_{j=i_k+1-1}^{n} (|a[j]| == x)$$

$$S_k = \sum_{j=i_k+1}^{n} (|a[j]| == x) + (a[i_k] == x)$$

$\Downarrow$

Since $a[i_k] \neq x$ from Conditions this returns Zero

$$\Rightarrow S_k = \sum_{j=i_{k+1}}^{n} (|a[j]| == x)$$

is true based on the assumption in II