(a)

lets say opt(i) be the length of longest possible path throught the sub tree with all vertives of the same colour.

for

and assume leftnode(T(i))  be opt(i-1) for the left node,rightnode(T(i)) be opt(i-1) of the right node

if their isn't node at that position, leftnode(T(i)) or rightnode(Ti) = -infinity.

let node(T(i)) be a array that stored the maximum length from smallest child node to T(i) node and the colour of this path

opt(0) = [1,T(0)]

opt(1) = MAX(all nodes linked to T(1))

node(T(1)) = [2,T(1)] if T(1) = T(0)

node(T(1)) = [-infinity,] if T(1) != T(0)

let them be the base case for all the leaves node.

then recursively:

we have

opt(i) = MAX(all nodes linked to T(i))

node(T(i)) = [opt(i-1)[0]+1,T(i)] if T(i) = T(i-1)

node(T(i)) = [-infinity,] if T(i) != T(i-1)

it takes O(4n) in the worst case which is also O(n)

(b)

all the others codition stay the same,but lets change the base case

create a variable Maximum = 0;

opt(0) = [1,T(0)]

opt(1) = MAX(all nodes linked to T(1)

node(T(1)) = [2,T(1)] if T(1) = T(0)

node(T(1)) = [0,T(1)] if T(1) != T(0) and check if  opt(0)[0] > Maximum, Maximum = opt(0)[0] ,O(1)

then recursively:

we have

opt(i) = MAX(opt(leftnode(Ti)),opt(rightnode(Ti)))

node(T(i)) = [opt(i-1)[0]+1,T(i)] if T(i) = T(i-1)

node(T(i)) = [0,T(i)] if T(i) != T(i-1)and check if  opt(i-1)[0] > Maximum, Maximum = opt(i-1)[0] ,O(1)

and at the very end , the answer will be MAX(Maximum,opt(i))

it takes O(4n) in the worst case which is also O(n)