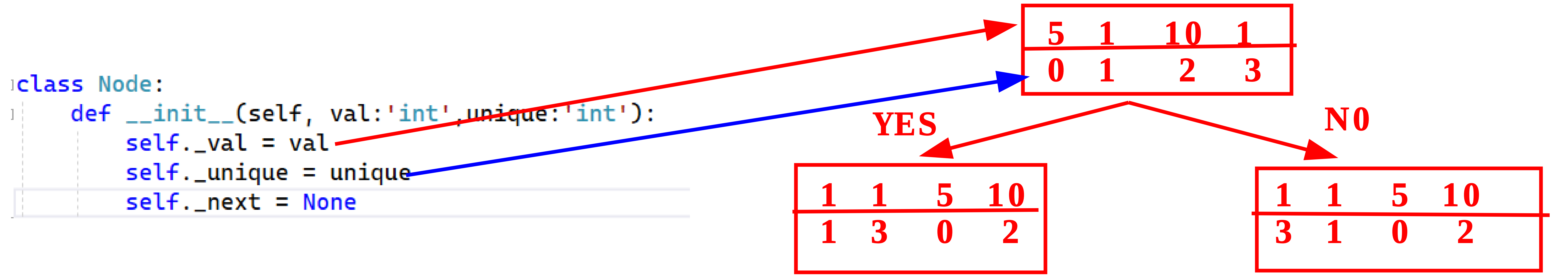
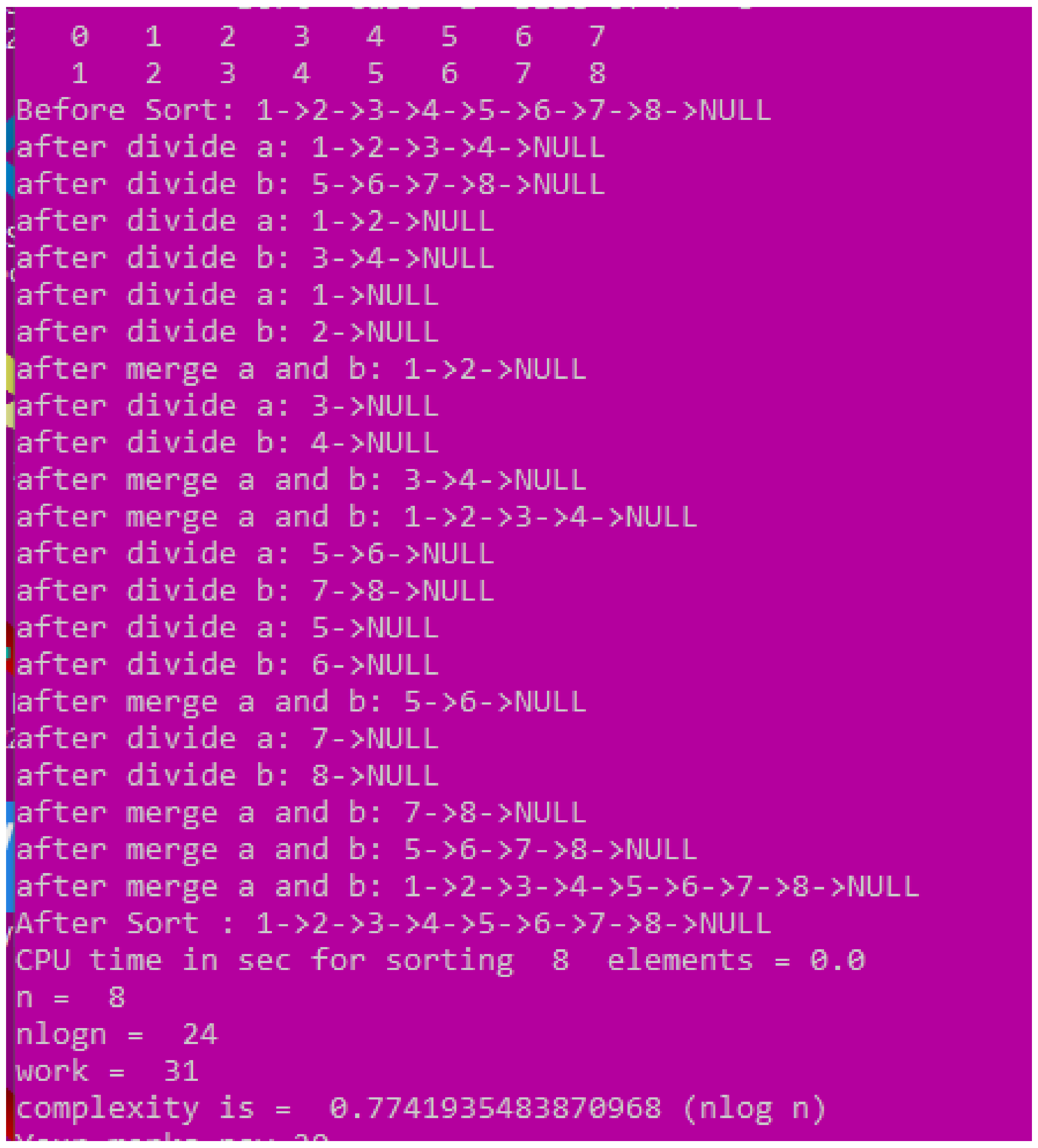
Inplace stable slist sort in ascending order

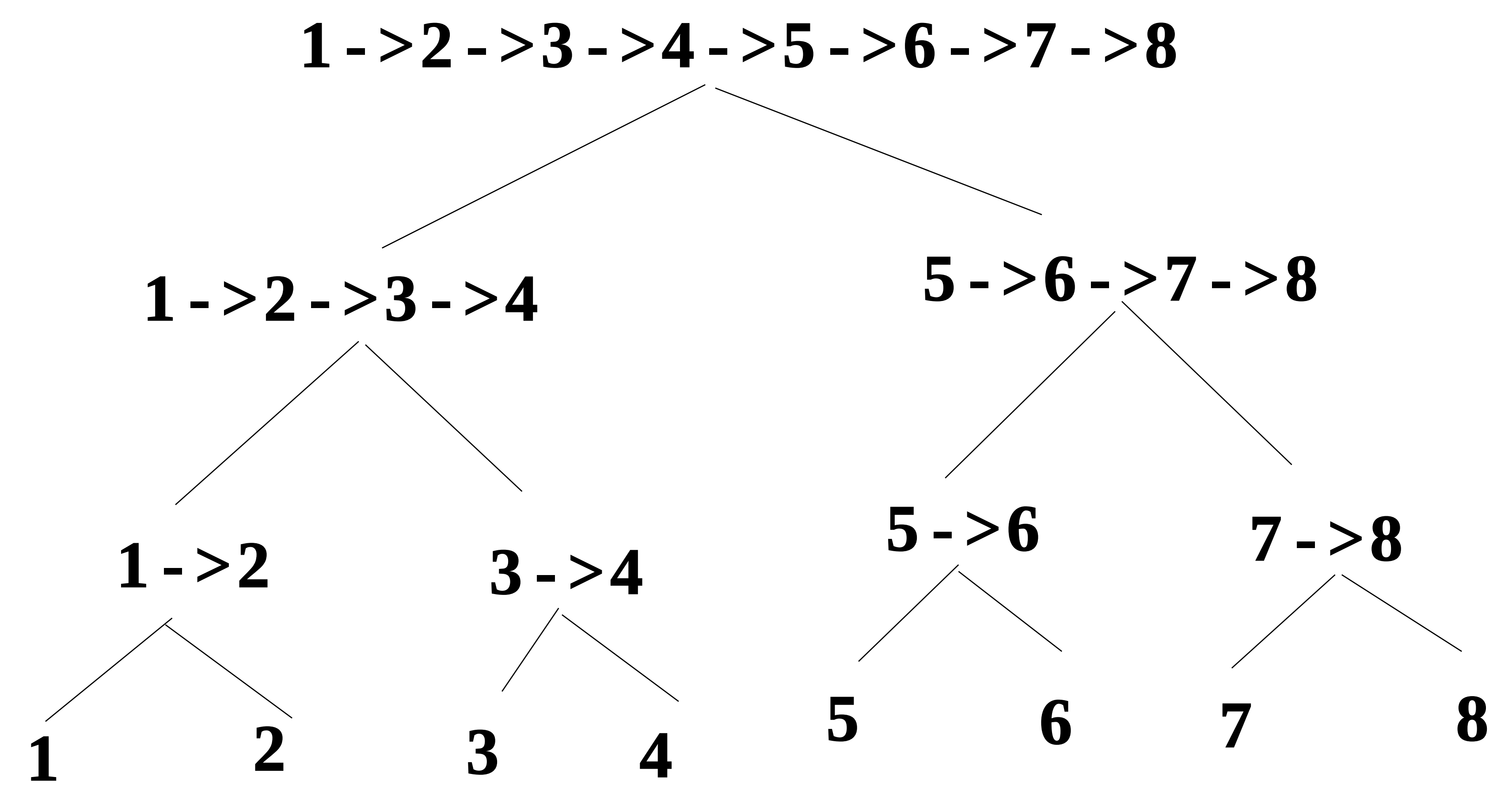


```
class Node:
    def __init__(self, val:'int',unique:'int'):
        self._val = val
        self._unique = unique
        self._next = None
```

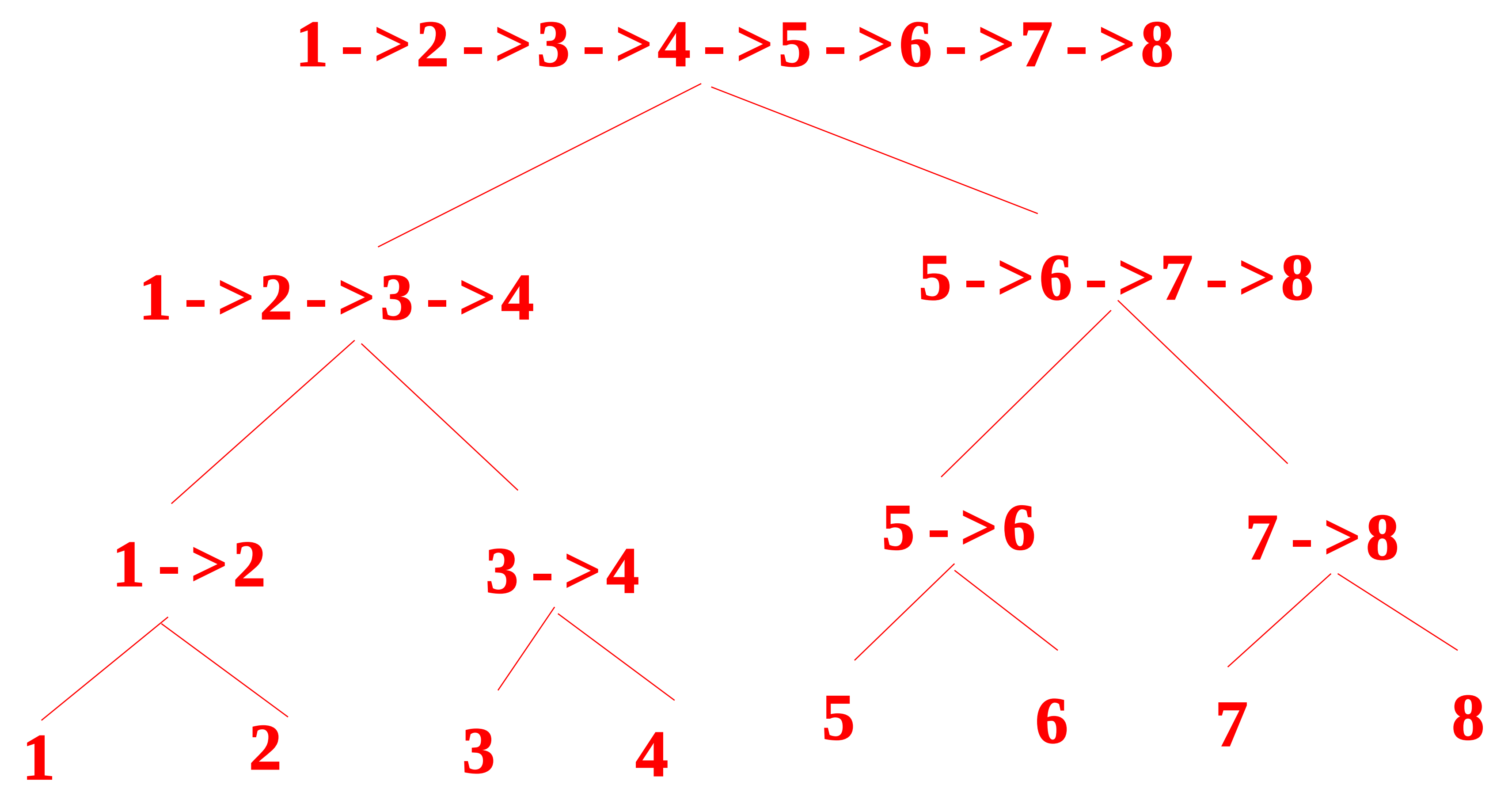
MUST PRINT ALL STEPS AS SHOWN BELOW FOR 100%



```
   0    1    2    3    4    5    6    7
   1    2    3    4    5    6    7    8
Before Sort: 1->2->3->4->5->6->7->8->NULL
after divide a: 1->2->3->4->NULL
after divide b: 5->6->7->8->NULL
after divide a: 1->2->NULL
after divide b: 3->4->NULL
after divide a: 1->NULL
after divide b: 2->NULL
after merge a and b: 1->2->NULL
after divide a: 3->NULL
after divide b: 4->NULL
after merge a and b: 3->4->NULL
after merge a and b: 1->2->3->4->NULL
after divide a: 5->6->NULL
after divide b: 7->8->NULL
after divide a: 5->NULL
after divide b: 6->NULL
after merge a and b: 5->6->NULL
after divide a: 7->NULL
after divide b: 8->NULL
after merge a and b: 7->8->NULL
after merge a and b: 5->6->7->8->NULL
after merge a and b: 1->2->3->4->5->6->7->8->NULL
After Sort : 1->2->3->4->5->6->7->8->NULL
CPU time in sec for sorting  8  elements = 0.0
n =  8
nlogn =  24
work =  31
complexity is =  0.7741935483870968 (nlog n)
Your marks now 20
```



1->2->3->4->5->6->7->8
1->2->3->4
5->6->7->8
1->2
3->4
5->6
7->8
1
2
3
4
5
6
7
8



1->2->3->4->5->6->7->8
1->2->3->4
5->6->7->8
1->2
3->4
5->6
7->8
1
2
3
4
5
6
7
8