CSCB63 TUT4 WEEK12
andrewyk.leung@mail.utoronto.ca
https://snoopysnipe.github.io/ta/b63w21/

1. Charge append \$4
   normal append uses \$1 save \$3

   n is array size
   m is # elements in array
   When $m = n$, create new array of size $1.5n$
   New array has $n/3$ empty slots
   Moving all elements over costs m
   After another $n/3$ appends array is full again, since saved \$3 per append, we still have at least \$m savings. which is enough to move them over again.

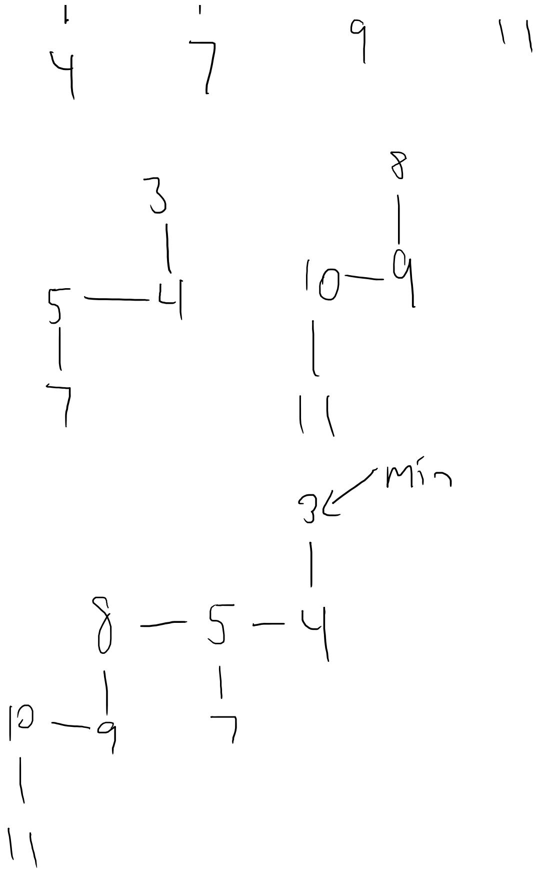
   So \$4 is sufficient to cover the costs
   So amortized complexity of append is $O(1)$.

2. a)

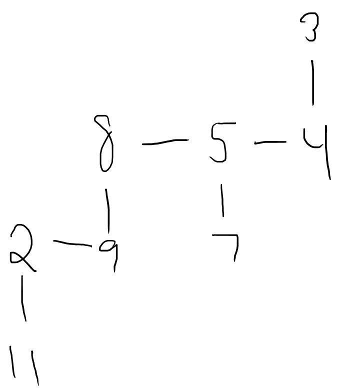
min
↓
2 - 3 - 4 - 5 - 7 - 8 - 9 - 10 - 11

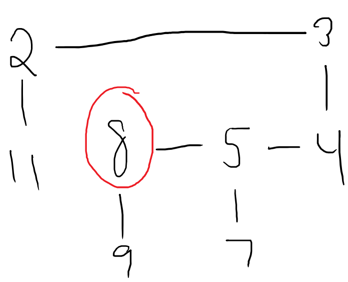
b)

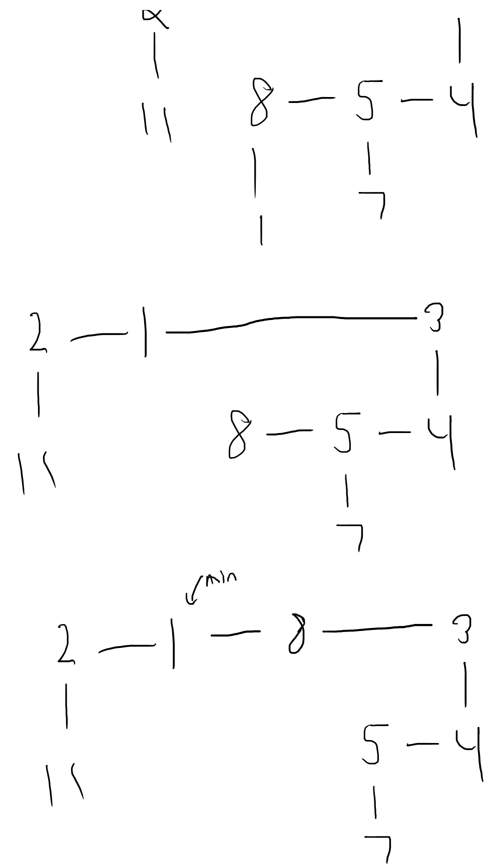
```
3    5    8    10
|    |    |    |
4    7    9    11
```



c)







3.
```
def has_route(i, j):
    for i=1 to n
        make_set(c_i)
    for each (c_i, c_j) ∈ R:
        Union(c_i, c_j)
    return find(c_i) == find(c_j)
```

4. Since the only difference is path compression, need a find operation. Path compression is only useful when finding a node that is

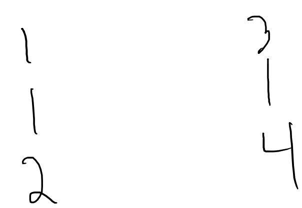
atleast 2 nodes away from root.

To make a tree of height 2, need to union 2 2-node trees.

eg: make(1), make(2), make(3), make(4)

1 2 3 4

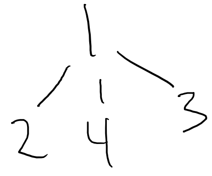
union(1,2) union(3,4)



Union(1,3)

← without path compression

now find deepest node, find(4)



← with path compression