A Taste Of Property-based Testing

Ruhi Choudhury

https://ruhi.dev

What-based testing?

For a given list of integers, return the maximum possible product of two elements in the list

Thanks to Hillel Wayne for this example

```
def max_prod(l: list[int]) -> int:
  if len(l) < 2:
    throw ValueError(“☹”)
  x, y = sorted(l, reverse=True)[0:2]
  return x * y
```

Thanks to Hillel Wayne for this example

```
def max_prod(l: list[int]) -> int:
 if len(l) < 2:
  throw ValueError(“☹”)
 x, y = sorted(l, reverse=True)[0:2]
 return x * y
```

Thanks to Hillel Wayne for this example

```
def max_prod(l: list[int]) -> int:
    if len(l) < 2:
        throw ValueError(“☹”)
    x, y = sorted(l, reverse=True)[0:2]
    return x * y
```

Thanks to Hillel Wayne for this example

```
def max_prod(l: list[int]) -> int:
  if len(l) < 2:
    throw ValueError(“☹”)
  x, y = sorted(l, reverse=True)[0:2]
  return x * y
```

Thanks to Hillel Wayne for this example

```
def test1():
  assert max_prod([1,2,5]) == 10
```

Thanks to Hillel Wayne for this example

```
def test2():
    assert max_prod([5,1,3,2]) == 15
```

Thanks to Hillel Wayne for this example

```
def test3():
  with raises(ValueError):
    max_prod([1])
```

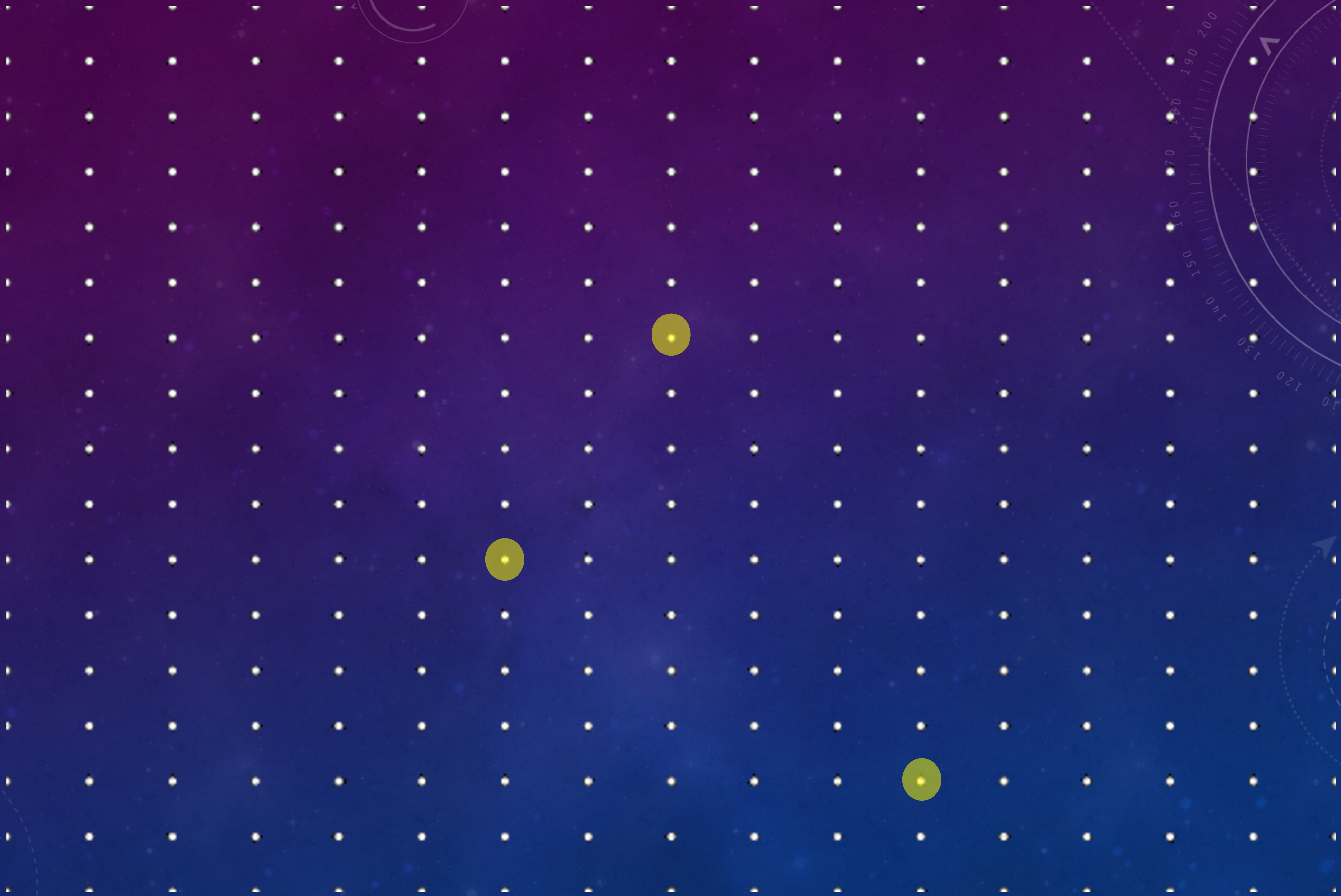
Thanks to Hillel Wayne for this example

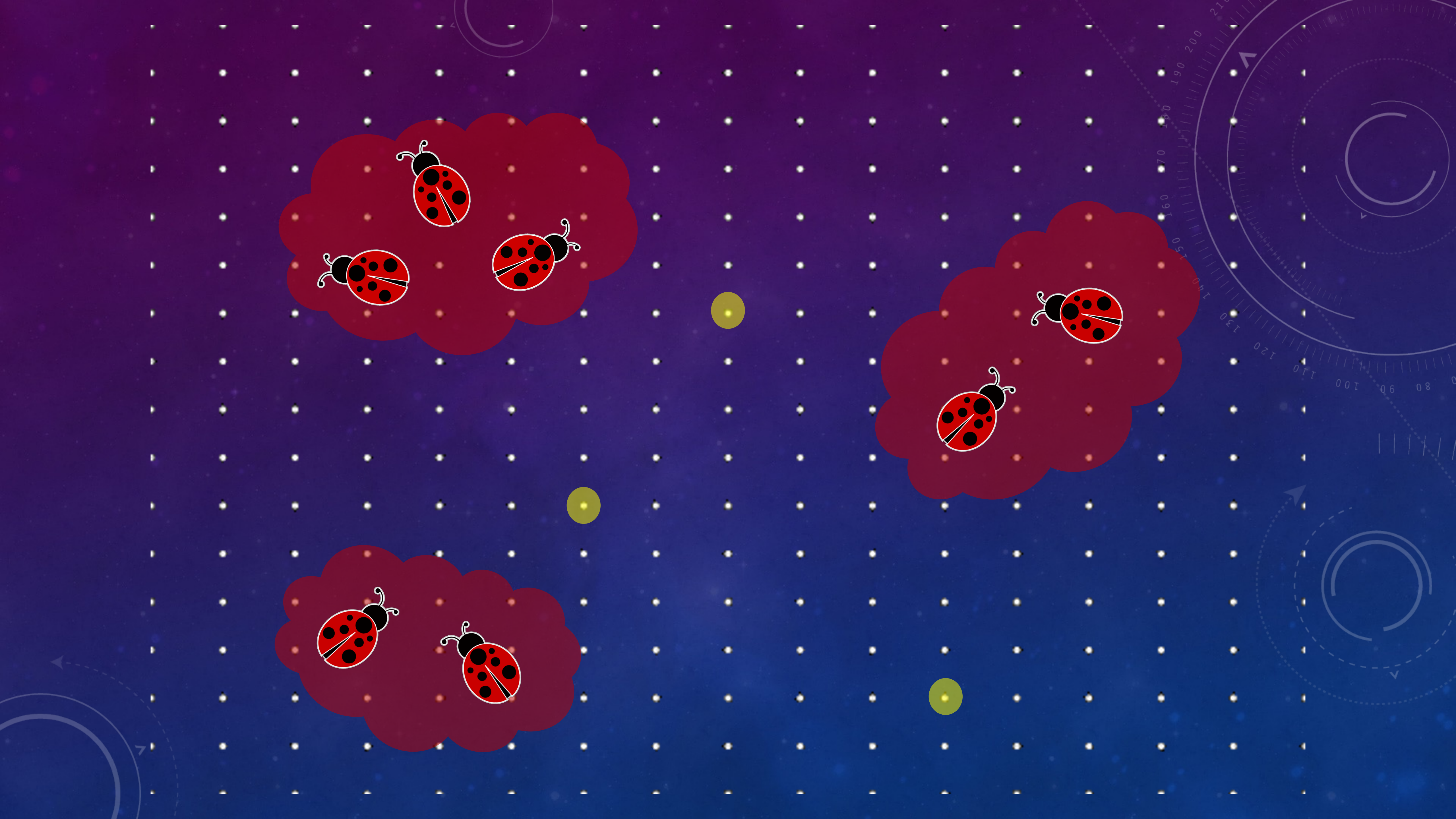
```
def test1():
    assert max_prod([1,2,5]) == 10

def test2():
    assert max_prod([5,1,3,2]) == 15

def test3():
    with raises(ValueError):
        max_prod([1])
```

Thanks to Hillel Wayne for this example





Randomly generate inputs

```
@given(l=lists(integers(), min_size=2))
def test_random(l):
  assert max_prod(l) == ???
```

```
@given(l=lists(integers(), min_size=2))
def test_random(l):
  assert max_prod(l) == ???
```

```
[5, 1, 3, 2]
```

5	1	1	2
5	2	1	3
5	3	2	3

5 × 1 = 5

5 × 2 = 10

5 × 3 = 15

1 × 2 = 2

1 × 3 = 3

2 × 3 = 6

$5 \times 1 = 5$

$5 \times 2 = 10$

$5 \times 3 = 15$

$1 \times 2 = 2$

$1 \times 3 = 3$

$2 \times 3 = 6$

```
@given(l=lists(integers(), min_size=2))
def test_random(l):
  all_pairs = itertools.combinations(l, 2)
  all_prods = [x * y for x,y in all_pairs]
  result = max_prod(l)
  assert all(result >= p for p in all_prods)
```

```
@given(l=lists(integers(), min_size=2))
def test_random(l):
  all_pairs = itertools.combinations(l, 2)
  all_prods = [x * y for x,y in all_pairs]
  result = max_prod(l)
  assert all(result >= p for p in all_prods)
```

```
@given(l=lists(integers(), min_size=2))
def test_random(l):
  all_pairs = itertools.combinations(l, 2)
  all_prods = [x * y for x,y in all_pairs]
  result = max_prod(l)
  assert all(result >= p for p in all_prods)
```

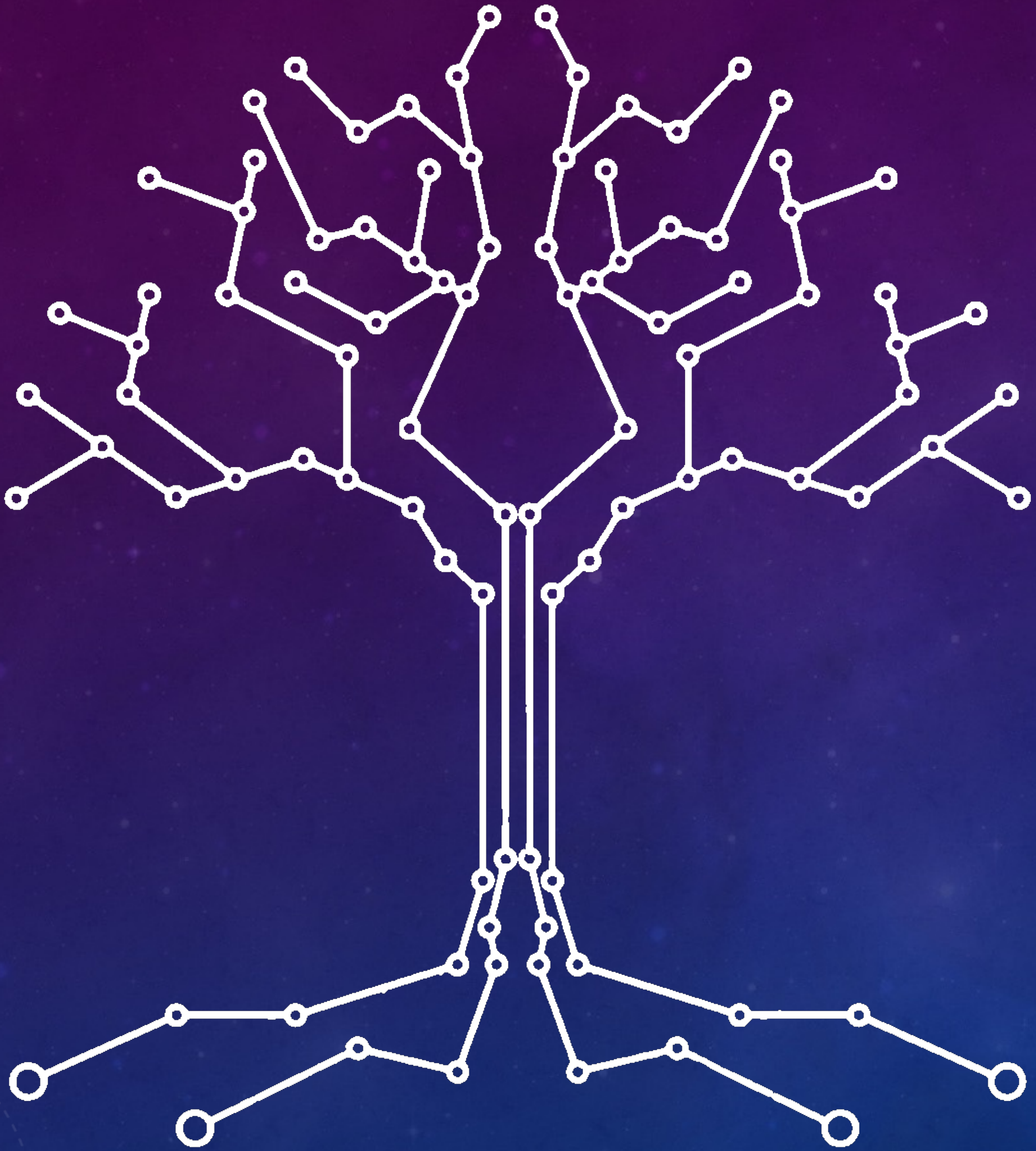
```
@given(l=lists(integers(), min_size=2))
def test_random(l):
    all_pairs = itertools.combinations(l, 2)
    all_prods = [x * y for x,y in all_pairs]
    result = max_prod(l)
    assert all(result >= p for p in all_prods)
```

```
Falsifying example:
  test_random(l = [-1, -1, 0])
E assert 0 >= 1
```

```
Falsifying example:
  test_random(l = [-1, -1, 0])
E assert 0 >= 1
```

our imp.

actual max. prod.



```
def insert(tree, key, value):
  # returns new tree with k-v pair
```

Thanks to John Hughes for this example

```
def delete(tree, key):
  # returns new tree without k-v pair
```

Thanks to John Hughes for this example

```
def find(tree, key):
  # returns value associated with key
```

Thanks to John Hughes for this example

```
def insert(tree, key, value):
  # returns new tree with k-v pair

def delete(tree, key):
  # returns new tree without k-v

def find(tree, key):
  # returns value associated with key
```

Thanks to John Hughes for this example

```
@given(tree=trees(), key=integers())
def test_find(tree, key):
    assert find(tree, key) == ???
```

Thanks to John Hughes for this example

```
@given(tree=trees(), key=integers())
def test_find(tree, key):
    assert find(tree, key) == ???
```

Thanks to John Hughes for this example

```
@given(t=trees(), k=integers(), v = integers())
def test_find1(t, k, v):
    treeWithKey = insert(t, k, v)
    assert find(treeWithKey, k) == v
```

Thanks to John Hughes for this example

```
@given(t=trees(), k=integers())
def test_find2(t, k):
    treeWithoutKey = delete(t, k)
    assert find(treeWithoutKey, k) == None
```

Thanks to John Hughes for this example

Ruhi Choudhury

https://ruhi.dev