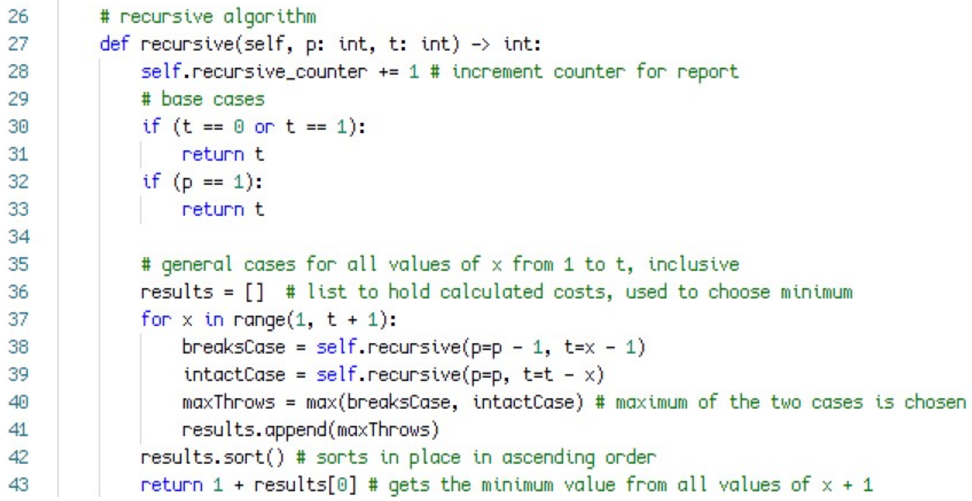
Ally Smith

Dynamic Programming Project

Recursive Implementation

Below is the code I used for my recursive implementation of the algorithm. When run with the parameters $p = 3,\ t = 16$ the function is called 753,665 times.



```python
# recursive algorithm
def recursive(self, p: int, t: int) -> int:
    self.recursive_counter += 1 # increment counter for report
    # base cases
    if (t == 0 or t == 1):
        return t
    if (p == 1):
        return t

    # general cases for all values of x from 1 to t, inclusive
    results = []  # list to hold calculated costs, used to choose minimum
    for x in range(1, t + 1):
        breaksCase = self.recursive(p=p - 1, t=x - 1)
        intactCase = self.recursive(p=p, t=t - x)
        maxThrows = max(breaksCase, intactCase) # maximum of the two cases is chosen
        results.append(maxThrows)
    results.sort() # sorts in place in ascending order
    return 1 + results[0] # gets the minimum value from all values of x + 1
```

Recursive Runtime Analysis (measured in ms)

$\frac{t}{p}$	10	12	15	18	20
2	1	2	20	170	716
4	7	39	499	5,942	29,772
8	13	117	2,958	65,222	
12	13	134	3,398		

As can be seen in the table above, the runtime increases greatly with slight changes in t. Additionally, higher values of p begin to affect the runtime more as t increases. For example, the runtime of $p = 8,\ t = 10$ is fairly close to that of $p = 4,\ t = 10$, but $p = 8,\ t = 15$ is far greater than $p = 4,\ t = 15$.