CSE 361 HW-3

Daniel Dyla

October 11, 2016

1 Problem Statement

Given a list of integers, return the 3 elements that produce the largest product.

2 Code

```
def read_dat(fname):
    with open(fname) as f:
        for l in f.readlines():
            yield([int(s.strip()) for s in l.split(',')])

def max_3prod(a):
    n = len(a)
    if n < 3:
        return
    ml = a[:3]
    m = a[0] * a[1] * a[2]
    for i in range(n-2):
        for j in range(i+1, n-1):
            for k in range(j+1, n):
                t = m
                m = max(m, a[i] * a[j] * a[k])
                if m > t:
                    ml = [a[i], a[j], a[k]]
    return ml, m
```

```
s = 0
for i in read_dat('sum-of-k.data'):
    lst, p = max_3prod(i)
    s += p
    print '[' + " ".join(str(i) for i in lst) + ']  -->', p

print "TOTAL:", s
```

3 Output

```
[1 2 3]  -> 6
[-1 -2 -3]  -> -6
[2 3 4]  -> 24
[2 3 4]  -> 24
[1 3 4]  -> 12
[1 2 4]  -> 8
[1 2 3]  -> 6
[-1 -2 4]  -> 8
[-1 -3 4]  -> 12
[-1 3 -4]  -> 12
[-2 -3 4]  -> 24
[-2 3 -4]  -> 24
[2 -3 -4]  -> 24
[-1 -2 -3]  -> -6
[1 -3 -4]  -> 12
[2 -3 -4]  -> 24
[-2 3 -4]  -> 24
[-2 -3 4]  -> 24
[3 4 5]  -> 60
[-1 -2 -3]  -> -6
[2 -4 -5]  -> 40
[3 4 5]  -> 60
[-2 -3 5]  -> 30
[-2 -4 5]  -> 40
[-3 4 -5]  -> 60
[2 3 4]  -> 24
[3 4 5]  -> 60
```

```
[-3 -4 5]  -> 60
[1 -4 -5]  -> 20
[5 6 7]  -> 210
[2 3 4]  -> 24
[2 3 4]  -> 24
[-2 -3 4]  -> 24
[2 -3 -4]  -> 24
[-1 -2 2]  -> 4
[-1 -2 1]  -> 2
[-1 -2 0]  -> 0
[-2 3 -3]  -> 18
[2 -4 -3]  -> 24
[0 -1 -2]  -> 0
[1 -2 -3]  -> 6
[-1 -2 3]  -> 6
TOTAL: 1070
```

4 Correctness

The algorithm terminates because each of the 3 for loops have pre-defined bounds.

The algorithm begins by assuming the 3 left-most elements of the array are the maximum product, and stores the product of those 3 elements. It then loops through every possible combination of 3 elements of the array, storing the combination and its product if the product of the combination is larger than the previous largest product. Every possible combination is checked, and each is checked against the previous maximum, so when the algorithm terminates the largest combination of 3 elements is returned. Thus the algorithm is correct.

5 Asymptotic Runtime

Lines 16-18 (inside the innermost for-loop) perform a constant amount of work (in the worst case) on each iteration, so it can be counted as a single operation.

$$= \sum_{i=0}^{n-3} \sum_{j=i+1}^{n-2} \sum_{k=j+1}^{n-1} 1 \tag{1}$$

$$= \sum_{i=0}^{n-3} \sum_{j=i+1}^{n-2} ((n-1) - (j+1) + 1) \tag{2}$$

$$= \sum_{i=0}^{n-3} \sum_{j=i+1}^{n-2} (n-1-j) \tag{3}$$

$$= \sum_{i=0}^{n-3} (\sum_{j=i+1}^{n-2} (n-1) - \sum_{j=i+1}^{n-2} j) \tag{4}$$

$$= \sum_{i=0}^{n-3} ((n-1)((n-2) - (i+1) + 1) - \sum_{j=i+1}^{n-2} j) \tag{5}$$

$$= \sum_{i=0}^{n-3} ((n-1)(n-2-i) - \sum_{j=i+1}^{n-2} j) \tag{6}$$

$$= \sum_{i=0}^{n-3} ((n^2 - 2n - in - n + 2 + i) - \sum_{j=i+1}^{n-2} j) \tag{7}$$

$$= \sum_{i=0}^{n-3} ((n^2 - 2n - in - n + 2 + i) - (\sum_{j=0}^{n-2} j - \sum_{j=0}^{i} j)) \tag{8}$$

$$= \sum_{i=0}^{n-3} ((n^2 - 2n - in - n + 2 + i) - (\sum_{j=0}^{n-2} j - \frac{i(i+1)}{2})) \tag{9}$$

$$= \sum_{i=0}^{n-3} ((n^2 - 2n - in - n + 2 + i) - (\frac{(n-2)(n-1)}{2} - \frac{i(i+1)}{2})) \tag{10}$$

$$= \sum_{i=0}^{n-3} ((n^2 - 2n - in - n + 2 + i) - (\frac{n^2 - 3n + 2}{2} - \frac{i^2 + i}{2})) \tag{11}$$

$$= \sum_{i=0}^{n-3} ((n^2 - 2n - in - n + 2 + i) - \frac{n^2 - 3n + 2 - i^2 - i}{2}) \tag{12}$$

$$= \sum_{i=0}^{n-3} (n^2 - 2n - in - n + 2 + i - n^2/2 + 3n/2 - 1 + i^2/2 + i/2) \tag{13}$$

$$= \frac{1}{6}(n-2)(n-1)n \tag{14}$$

$$\in \Theta(n^3) \tag{15}$$