

May 5, 2016

0.0.1 Problem (Bentley)

Given numbers $a_1 \dots a_n$, find a contiguous subsequence ("block") s.t. largest sum.

```
In [1]: #  $O(N^3)$ 
def largest_block_sum1(numbers):
    best = 0
    n = len(numbers)
    for i in range(n):
        for j in range(i+1, n):
            total = 0
            for k in range(i, j):
                total += numbers[k]
            best = max(best, total)
    return best

#  $O(N^2)$ 
def largest_block_sum2(numbers):
    best = 0
    n = len(numbers)
    for i in range(n):
        total = 0
        for j in range(i, n):
            total += numbers[j]
            best = max(best, total)
    return best

#  $O(N \log N)$ 
# divide and conquer
def largest_block_sum3(numbers):
    if len(numbers) == 0:
        return 0
    elif len(numbers) == 1:
        return max(0, numbers[0])

    n = len(numbers)
    midpoint = n // 2
    subproblems = max(largest_block_sum3(numbers[:midpoint]),
```

```

        largest_block_sum3(numbers[midpoint:])) # copies 'ca

right = 0
current = 0
for i in range(midpoint, n):
    current += numbers[i]
    right = max(right, current)

left = 0
current = 0
for i in range(midpoint - 1, -1, -1):
    current += numbers[i]
    left = max(left, current)

return max(subproblems, left + right)

# O(N)
def largest_block_sum4(numbers):
    best = 0
    current = 0
    for number in numbers:
        current = max(current + number, number)
        best = max(best, current)
    return best

In [2]: test_array = [1, -6, 3, -1, 4, 2, -3, 2]

assert largest_block_sum1(test_array) == 8
assert largest_block_sum2(test_array) == 8
assert largest_block_sum3(test_array) == 8
assert largest_block_sum4(test_array) == 8

```