

(a) We prove this for  $f(n) = n^3$ . The outer loop of the given algorithm runs for exactly  $n$  iterations, and the inner loop of the algorithm runs for at most  $n$  iterations every time it is executed. Therefore, the line of code that adds up array entries  $A[i]$  through  $A[j]$  (for various  $i$ 's and  $j$ 's) is executed at most  $n^2$  times. Adding up array entries  $A[i]$  through  $A[j]$  takes  $O(j - i + 1)$  operations, which is always at most  $O(n)$ . Storing the result in  $B[i, j]$  requires only constant time. Therefore, the running time of the entire algorithm is at most  $n^2 \cdot O(n)$ , and so the algorithm runs in time  $O(n^3)$ .

(b) Consider the times during the execution of the algorithm when  $i \leq n/4$  and  $j \geq 3n/4$ . In these cases,  $j - i + 1 \geq 3n/4 - n/4 + 1 > n/2$ . Therefore, adding up the array entries  $A[i]$  through  $A[j]$  would require at least  $n/2$  operations, since there are more than  $n/2$  terms to add up. How many times during the execution of the given algorithm do we encounter such cases? There are  $(n/4)^2$  pairs  $(i, j)$  with  $i \leq n/4$  and  $j \geq 3n/4$ . The given algorithm enumerates over all of them, and as shown above, it must perform at least  $n/2$  operations for each such pair. Therefore, the algorithm must perform at least  $n/2 \cdot (n/4)^2 = n^3/32$  operations. This is  $\Omega(n^3)$ , as desired.

(c) Consider the following algorithm.

```

For  $i = 1, 2, \dots, n$ 
  Set  $B[i, i + 1]$  to  $A[i] + A[i + 1]$ 
For  $k = 2, 3, \dots, n - 1$ 
  For  $i = 1, 2, \dots, n - k$ 
    Set  $j = i + k$ 
    Set  $B[i, j]$  to be  $B[i, j - 1] + A[j]$ 

```

This algorithm works since the values  $B[i, j - 1]$  were already computed in the previous iteration of the outer for loop, when  $k$  was  $j - 1 - i$ , since  $j - 1 - i < j - i$ . It first computes  $B[i, i + 1]$  for all  $i$  by summing  $A[i]$  with  $A[i + 1]$ . This requires  $O(n)$  operations. For each  $k$ , it then computes all  $B[i, j]$  for  $j - i = k$  by setting  $B[i, j] = B[i, j - 1] + A[j]$ . For each  $k$ , this algorithm performs  $O(n)$  operations since there are at most  $n$   $B[i, j]$ 's such that  $j - i = k$ . There are less than  $n$  values of  $k$  to iterate over, so this algorithm has running time  $O(n^2)$ .

---

<sup>1</sup>ex474.221.961