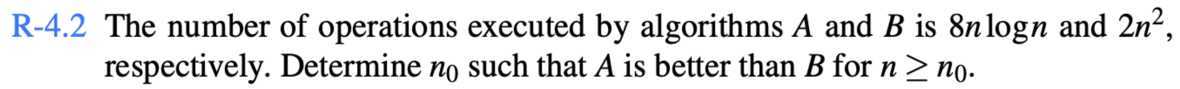
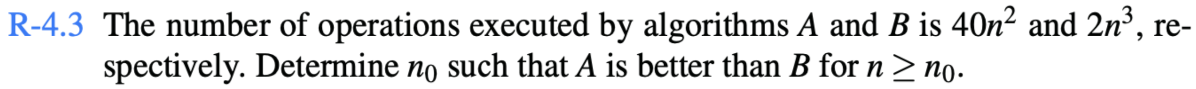
Recitation 4

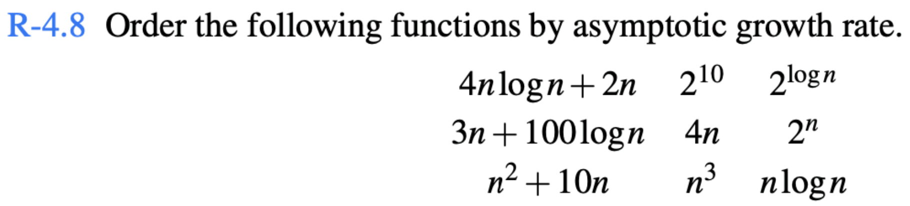
Asymptotic Analysis Practice



R-4.2 The number of operations executed by algorithms A and B is $8n\log n$ and $2n^2$, respectively. Determine n_0 such that A is better than B for $n \geq n_0$.



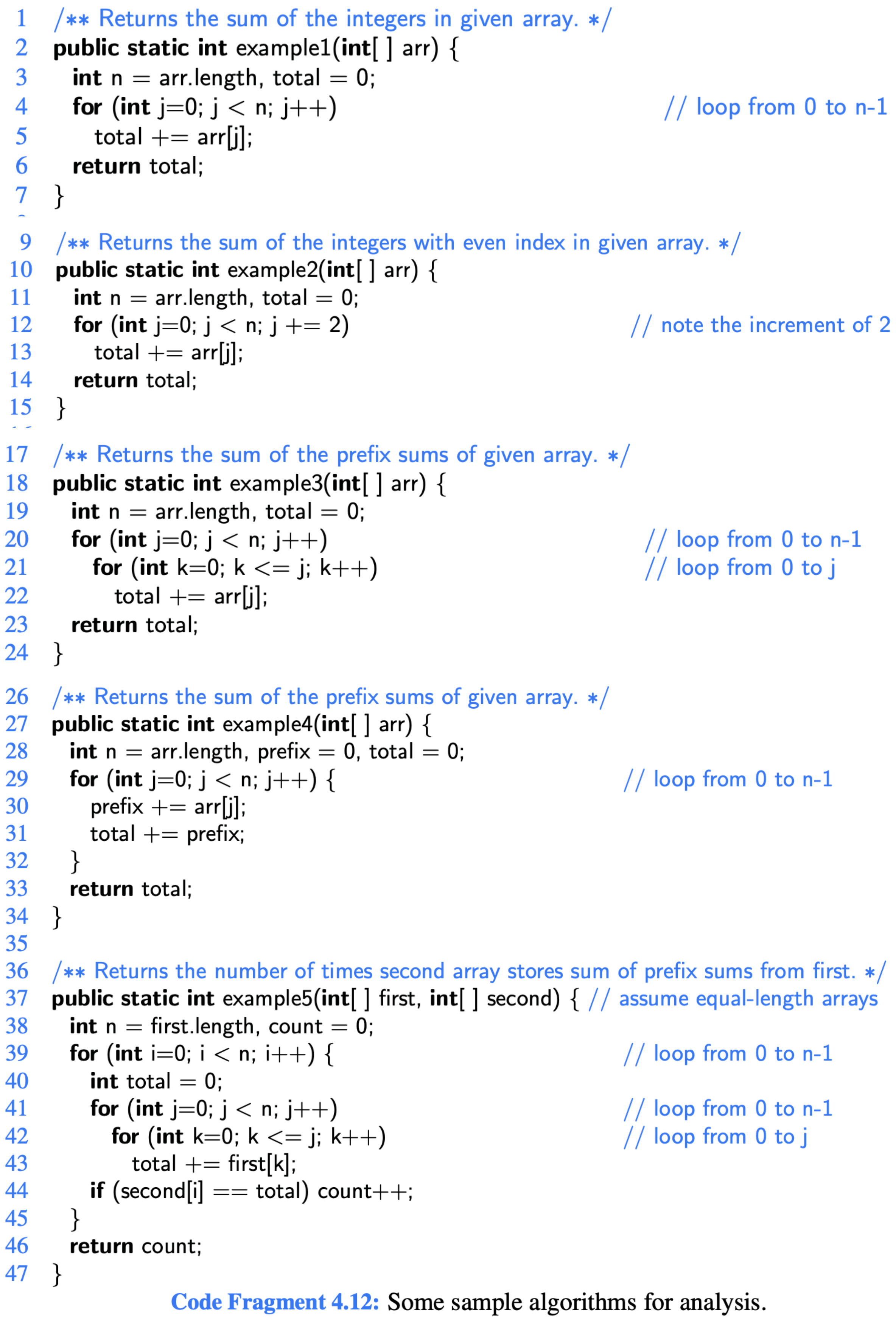
R-4.3 The number of operations executed by algorithms A and B is $40n^2$ and $2n^3$, respectively. Determine n_0 such that A is better than B for $n \geq n_0$.



R-4.8 Order the following functions by asymptotic growth rate.

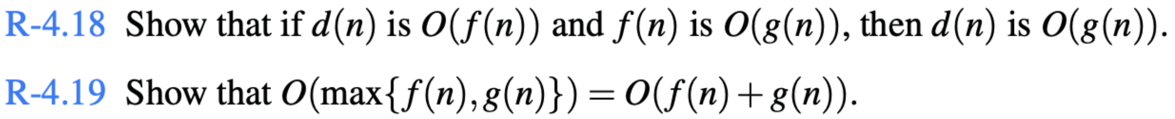
$$\begin{array}{ccc} 4n\log n + 2n & 2^{10} & 2^{\log n} \\ 3n + 100\log n & 4n & 2^n \\ n^2 + 10n & n^3 & n\log n \end{array}$$

Q: What is the asymptotic complexity for the following examples



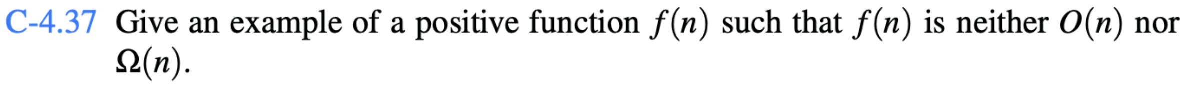
```
/** Returns the sum of the integers in given array. */
public static int example1(int[ ] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j++)                              // loop from 0 to n-1
    total += arr[j];
  return total;
}

/** Returns the sum of the integers with even index in given array. */
public static int example2(int[ ] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j += 2)                           // note the increment of 2
    total += arr[j];
  return total;
}

/** Returns the sum of the prefix sums of given array. */
public static int example3(int[ ] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j++)                              // loop from 0 to n-1
    for (int k=0; k <= j; k++)                           // loop from 0 to j
      total += arr[j];
  return total;
}

/** Returns the sum of the prefix sums of given array. */
public static int example4(int[ ] arr) {
  int n = arr.length, prefix = 0, total = 0;
  for (int j=0; j < n; j++) {                            // loop from 0 to n-1
    prefix += arr[j];
    total += prefix;
  }
  return total;
}

/** Returns the number of times second array stores sum of prefix sums from first. */
public static int example5(int[ ] first, int[ ] second) { // assume equal-length arrays
  int n = first.length, count = 0;
  for (int i=0; i < n; i++) {                            // loop from 0 to n-1
    int total = 0;
    for (int j=0; j < n; j++)                            // loop from 0 to n-1
      for (int k=0; k <= j; k++)                         // loop from 0 to j
        total += first[k];
    if (second[i] == total) count++;
  }
  return count;
}
```

Code Fragment 4.12: Some sample algorithms for analysis.



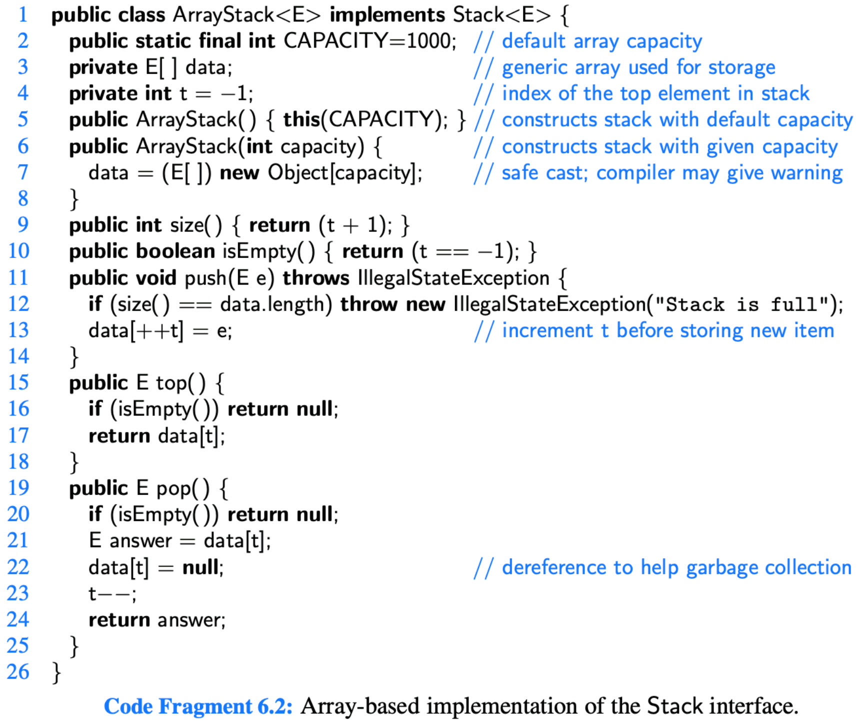
R-4.18 Show that if $d(n)$ is $O(f(n))$ and $f(n)$ is $O(g(n))$, then $d(n)$ is $O(g(n))$.

R-4.19 Show that $O(\max\{f(n), g(n)\}) = O(f(n) + g(n))$.



C-4.37 Give an example of a positive function $f(n)$ such that $f(n)$ is neither $O(n)$ nor $\Omega(n)$.

Another implementation for stack

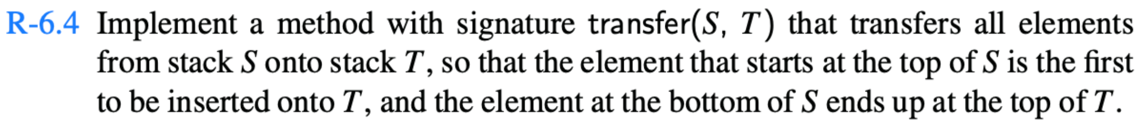


```
public class ArrayStack<E> implements Stack<E> {
  public static final int CAPACITY=1000;  // default array capacity
  private E[ ] data;                      // generic array used for storage
  private int t = −1;                     // index of the top element in stack
  public ArrayStack( ) { this(CAPACITY); } // constructs stack with default capacity
  public ArrayStack(int capacity) {       // constructs stack with given capacity
    data = (E[ ]) new Object[capacity];   // safe cast; compiler may give warning
  }
  public int size( ) { return (t + 1); }
  public boolean isEmpty( ) { return (t == −1); }
  public void push(E e) throws IllegalStateException {
    if (size( ) == data.length) throw new IllegalStateException("Stack is full");
    data[++t] = e;                        // increment t before storing new item
  }
  public E top( ) {
    if (isEmpty( )) return null;
    return data[t];
  }
  public E pop( ) {
    if (isEmpty( )) return null;
    E answer = data[t];
    data[t] = null;                       // dereference to help garbage collection
    t−−;
    return answer;
  }
}
```

Code Fragment 6.2: Array-based implementation of the Stack interface.

Q: What are the advantages and disadvantages of this implementation, compared with the linked list version seen in class?

Stack and Queue Exercise



R-6.4 Implement a method with signature transfer(S, T) that transfers all elements from stack S onto stack T, so that the element that starts at the top of S is the first to be inserted onto T, and the element at the bottom of S ends up at the top of T.

Q: What if I want the elements starting at the top of S ends up at the top of T?