#### Running Time Analysis

Zhuo Zhang zzhang90@bu.edu

1.

public static int example1(int[] arr) {

int n = arr.length, total = 0; // c1

for (int j=0; j < n; j++) // list is iterated once   n

total += arr[j]; // c2

return total;

}

Total run time: c1 + c2 \* n = O(n)

2.

public static int example2(int[] arr) {

      int n = arr.length, total = 0; // c1

      for (int j=0; j < n; j += 2) // list iterated once skipping every other one n/2

            total += arr[j]; // c2

      return total;

}

Total run time: c1 + c2 \* n / 2 = O(n)

3.

 public static int example3(int[] arr) {

      int n = arr.length, total = 0; // c1

      for (int j=0; j < n; j++) // list iterated once n

              for (int k=0; k <= j; k++) // nested list iterated once n \* n

               total += arr[j]; c2

      return total;

}

Total run time: c1 + c2 \* n \* n = O(n^2)

4.

 public static int example4(int[] arr) {

      int n = arr.length, prefix = 0, total = 0; // c1

      for (int j=0; j < n; j++) { // list iterated once n

             prefix += arr[j]; // c2

        total += prefix; // also run n times

      }

      return total;

 }

Total run time: c1 + c3 \* n + n = O(n)

5.

public static int example5(int[] first, int[] second) { // assume equal‐length arrays

int n = first.length, count = 0;      // c1

for (int i=0; i < n; i++) {         // n

int total = 0;        // c2

for (int j=0; j < n; j++)        // nested n \* n

for (int k=0; k <= j; k++)        // nested n \* n \* n

total += first[k];        // c3

if (second[i] == total) count++;      // c4 run once every time second for runs n \* n

}

return count;

}

Total run time: c1 + c2 \* n \* n + c3 \* n \* n \* n + c4 \* n \* n = O(n^3)