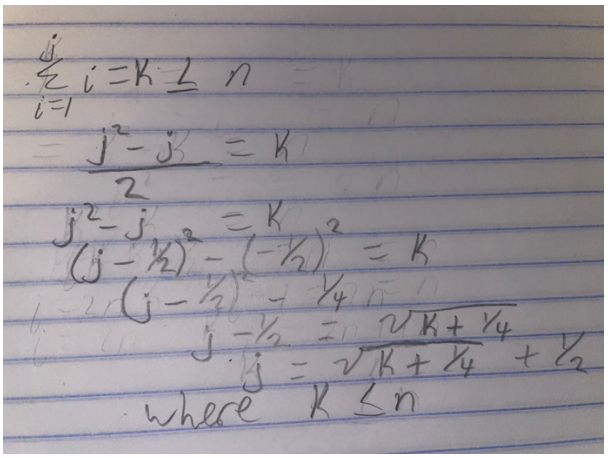
Nick Wurzer
V00958568

CSC225 A3 CountInversions Explanation and Runtime Analysis

For the worst case runtime analysis I will consider only assignments (A) and comparisons (C) for an array of length n and maximum inversions k = n.

```
public static int CountInversions(int[] A){
      int count = 0;                1A
      int i = 0;                    1A
      int j;
      int temp;
      //Loop through the whole array until an unordered element is found
      while(i < A.length - 1){      1C * n + 1
         if(A[i+1] < A[i]){         1C * n
            j = I;                  1A * (sqrt(k+1/4) + 1/2)    (see note at the end for runtime
            temp = A[j+1];          1A  * (sqrt(k+1/4) + 1/2)   regarding these two statements)
            /*loop towards start of array from unordered element until you find the spot it
belongs
             *bump every element up one spot as you loop backwards
             *increment the inversion count every time you bump up an element -this is an
inversion
             */
            while((j > 0) && A[j] > temp){   2C  * k + 1
               A[j+1] = A[j];               1A * k
               count++;                     1A * k
               j--;                         1A * k
            }
            //put the element that was out of order at the index which it belongs
            A[j+1] = temp;                  1A * (sqrt(k+1/4) + 1/2) (see note)
         }
         i++;                               1A * (sqrt(k+1/4) + 1/2)  (see note)
      }
      return count;                         1A
   }
```



The picture above is regarding the note on runtime. We know that the nested while loop will only run k times because there will only be k swaps, and every time that the while loop runs, we do a swap. For the worst case run time, with k = n swaps, the swaps will be located as close to the beginning as they can be, meaning the beginning of the array is sorted in the wrong way (highest to lowest). So everything inside the if statement will run j times, until the sum from 1 to j of I equals k. The picture indicates that j is bound by the square route of k.

Now we might think that we must multiply the second loop by the square route of k term after our initial calculation, but we already know that that nested while loop will run at most k times total (although k+1 comparisons will be made).

We now use the assignments and comparisons from above to calculate the worst case runtime analysis as shown.