Arpan Patel
CSC 321
Homework 3 Written answers

Below you will find pseudo-code for an algorithm that counts the number of pairs of identical values in an array. In analyzing the running tome for it, we decide to count how often the equality test `if a[i] == a[j]` is executed for an array of length *n*. Please:

a. Write down the double sum arising from the nested loops.

Answer:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \sum_{i=1}^{n} (n-i+1)$$

b. Determine and write down the closed-form arithmetic formula for that double sum.

Answer: (n^2/2) + (n/2)

```
countIdenticalPairs(a) {
  n = a.size
  count = 0
  for i = 1 to n {
    for j = i to n {
      if a[i] == a[j] {
        count += 1
      }
    }
  }
  return count
}
```

Realizing that the first time through the inner loop each time, we are comparing a value to itself and so counting it (and we shouldn't), we change the lower index on the inner loop. Once again, for this algorithm, please:

a. Write down the double sum arising from the nested loops.

Answer:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n} (n-(i+1)+1 = \sum_{i=1}^{n} (n-i)$$

b. Determine and write down the closed-form arithmetic formula for that double sum.

Answer: (n^2/2)-(n/2)

```
countIdenticalPairs(a) {
  n = a.size
  count = 0
  for i = 1 to n {
    for j = i+1 to n {
      if a[i] == a[j] {
        count += 1
      }
    }
  }
  return count
}
```

Runtime analysis for recursion (20 points)

Below you will find pseudo-code for an algorithm that finds the maximum value in an array. In analyzing the running tome for it, we decide to count how often the comparison `if leftmax > rightmax` is executed for an array of length *n*. Please:

a. Write down the recurrence relation arising from the recursive calls.

Answer:T(n)= 2T(n/2)+1

b. Determine and write down the closed-form arithmetic formula for that recurrence. I strongly suggest using the Main Recurrence Theorem in the book.

Answer: n^(log2(2))

```
findMax(a, i, j) {
  if i == j {
    return a[i]
  }
  mid = (i + j) / 2
  leftmax = findMax(a, i, mid)
  rightmax = findMax(a, mid+1, j)
  if leftmax > rightmax {
    return leftmax
  } else {
    return rightmax
  }
}
```

Counting inversions (35 points)

For an array of values, an inversion is a pair of numbers that are in the wrong order relative to a sorted array. For example, if `a = [4,3,5,2,1]`, the pair of values 4 at index 1 and 2 at index 4 are an inversion. For example, the array a has 8 inverted pairs: (4,3), (4,2), (4,1), (3,2), (3,1), (5,2), (5,1), (2,1).

a. How many inversions are there in the array [n, n-1, n-2, ..., 2, 1]? Explain your answer.

   Answer: (n+1)+(n+2)+…+1 = n(n-1)/2 This is the answer because every new element is less than the previous by one, thus as you compare the numbers through the array the amount of numbers printed will keep decreasing by one.

b. Implement a (simple) algorithm (that is, write a program) that counts the number of inversions in a list. Use the result from (a) to test your algorithm.

   Answer: Please look at the file ArpanPatel_Count_Inversion_Pairs.py

c. What is the asymptotic running time of your algorithm?

Answer: O(n^2)