Homework 2

Matt Forbes

October 21, 2010

1 Problem One

a)

found is true. For this case to occur, we had to be within some iteration of the loop. In order to be in an interation, the expression (low ¡ high) AND !found must be true. Therefore, low does not equal high, because low must be less than high. The first if statement within the loop body checks if
$A[low] + a[high] == x$ is true, and will set found to true if that is the case. Therefore, if found is true, then low and high are indices of A whose elements' sum is x.

b)

low $\geq$ **high**. I will prove the loop invariant provided to show that if low $\geq$ high, there does not exist two disting elements in **A** that sum to x.

<u>Basis</u>

> At the start of the first iteration,
> $low = 0, high = n - 1,$
> $S = \{A[0] \dots A[-1]\} \cup \{A[n] \dots A[n-1]\} = \{\}$
> So there are no distinct pairs in S sum to x. The L.I. holds before we enter the loop.

<u>Maintenance</u>

Assuming the L.I. held for all iterations up to this iteration j, then:

- Right before this loop started, there was no distinct pair of elements in the set $S = \{A[0] \dots A[low-1]\} \cup \{A[high+1] \dots A[n-1]\}$ whose sum = x.
- During this iteration, $A[low] + a[high]$ could be:
  equal to x: We found a distinct pair that sums x, done.

  less than x: low is incremented, and thus $A[low]$ is 'added' to S in the next iteration, in which case the L.I. would still hold for these reasons: $A[low] + A[i], i = 0 \dots low - 1$ will always be less than x, and $A[low] + a[j], j =$

$high + 1 \ldots n - 1$ will always be greater than x. Therefore there will be no pair in S whose sum is exactly x.

greater than x: high is decremented, and this $A[high]$ is 'added' to S in the next iteration, in which case the L.I. would still hold for these reasons: $A[high] + A[j], j = high + 1 \ldots n - 1$ will always be greater than x, and $A[i] + A[high], i = 0 \ldots low$ will always be less than x. Therefore there will be no pair in S whose sum is exactly x.

- The L.I. will always hold after this iteration, granted that it held up to this point for the reasons listed above.

Termination

- After each iteration, either low is incremented or high is decremented, so they have to converge at one point as long as **found** is never set to true, so the loop is guaranteed to terminate.
- According to the L.I. at the end of the last iteration, there is no distinct pair of elements in $S = \{A[0] \ldots A[low - 1]\} \cup \{A[high + 1] \ldots A[n - 1]\}$ whose sum is x. Well at the end of the loop, S is equal to all of the elements in **A**. Therefore, there is no distinct pair of elements in **A** whose sum is equal to x.

2 Problem Two

2.1 a)

(1,5) (2,5) (3,4) (3,5) (4,5)

2.2 b)

The array [n ... 1] has the most inversions.
A[1] is greater than n-1 elements on its right, so it has n-1 inverions.
A[2] is greater than n-2 elements on its right, so it has n-2 inversion.
...
A[n-1] is grater than n element on its right so it has 1 inversion.
So, the number of inversions: $\sum_{i=1}^{n-1} = \frac{n(n-1)}{2}$

2.3 c)

The number of writes insertion sort performs is equal to the number of inversions there were in the intial form of array. Each inversion in the array implies a shift of an element to the left.

2.4 d)

3 Problem Three

Algorithm Description

The algorithm's correctness stems from the following point being true: If we take the median card from both friends' hand, then the absolute median's value must be within the value of these two cards.

If we take the median card from both hands, and call card **a** the smaller of the two, and **b** the greater. There are at most $n - \frac{n}{2}$ cards greater than **a** in its own hand. In **b**'s hand, there are at most $n - \frac{n}{2} - 1$ cards greater than **a**, because **a** > **b**. In total, there are at most $n - \frac{n}{2} + n - \frac{n}{2} - 1 = 2n - n - 1 = n - 1$ cards greater than **a**. Therefore, **a** must be greater than or equal to the absolute median of the 2n cards.

Likewise for **b**.

In the algorithm, **L** and **R** represent the two hands of cards. **lbound** and **rbound** are the minimum and maximum values of the absolute median that we know up to that point. **cuts** is the number of cards that we have ruled out and know are less than the absolute median.

The pseudo code for my algorithm is on the last page.

Running time of algorithm

Let T(n) be the running time of this algorithm, where n is the total number of cards we are searching through.

$$T(n) = \left\{ \begin{array}{cc} T(\frac{n}{2}) + O(1), & \text{for n} > 2 \\ \text{O(1)}, & \text{for n} \leq 2 \end{array} \right\}$$

An upper bound guess for T is $T(n) \leq c\log_2(n) + O(1)$

$$\begin{aligned} T(n) &\leq c\log_2(\frac{n}{2}) + O(1) \\ &\leq c\log_2(n) - \log_2(2) + O(1) \\ &= c\log_2(n) + O(1) \end{aligned}$$

So, $T(n)$ is $O(\log_2(n))$.