Braden Ackles (932-115-298), Mihai Dan (932-146-380), Dylan Camus (931-769-466)

Group Project 2

11/18/2016

**Introduction:**

With given keys, find the shortest path to open all the lockers that contain tennis balls. There are *m* keys given initially, to open *n* lockers that contain *t* tennis balls. Each locker contains the keys to the adjacent lockers. Meaning locker *i* would contain the keys for lockers *i-1* and *i+1*.

**Algorithm 1 - Enumeration (Brute Force):**

**Pseudocode:**

For i=0 to number of lockers

If x is less than number of tennis balls

If balls array at location x-1 equals i

Then lockers array location i equals one

Increment x

For i=1 to number of keys plus one

Loop through the keys array after its been sorted

Set the counter to zero

Set the segments equal to the list function

If the iterator is equal to one

Set the total equal to the last ball location minus the first +1

If there is a ball but no key

Set total equal to balls array minus key array

Else if there is a key for the last ball location

Set the total equal to key array minus balls array at last location

Set the counter equal to total

Else

Set the counter to i

For j=0 to the length of the keyset array minus one

Append the segments at j and j+1 in the keyset array

For every pair found in these segments

Increment count by either pair or lockers depending on evalSegment

Increase count based on the result of countLeft with arrays set to zero

Increase count based on the result of countRight with arrays set to zero

Decide which is less by doing the minimum between minLockers and count

Return minLockers

**Runtime Analysis:**

* The run time of the algorithm is O(). This is because the algorithm goes through every locker, as well as every combination possible for each key. It starts out by doing the possible combinations for key 1, then moves on to key 2, and so on. After getting this information, it will evaluate two numbers, the new cost and previous cost, and if the new cost is less than the previous, it will replace it as the preferred route.

**Solutions:**

* **Set #1:** 11
* **Set #2:** 14
* **Set #3:** 7
* **Set #4:** 14
* **Set #5:** 19
* **Set #6:** 1
* **Set #7:** 15
* **Set #8:** 8

**Algorithm 2 - Dynamic Programming:**

**Pseudocode:**

If the key is the first key then

The total number of balls is checked to the left

If the first key is less than the first ball

Then add to the total balls

Else if the first key is greater than the last ball

Then add to the total balls

If there is a ball to the left of the first key

Add to the total number of balls

While there are keys to the left of the key

Go to the next position

Set the left key

While loop

Check if all balls are collected

If so return

Check if any balls left in the segment

Check left

Check right

If the number from right is less than from the left

Go from the left

Count the number of balls left in the segment

Add to the counter to track them

Return them

Add the collected balls to the collected list

If not last segment

Compare if close to get from left or right key

Increment counter

If it’s the last segment check for balls after it

You will only be coming from the left in this case

Add the left counter till find the balls

If no balls left in segment go to the next

**Runtime Analysis:**

* The pseudo code and actual code splits the keys in to segments, from the first locker to the first key from the first key to the second, from the I key to the I+1 key and from the last key to the N locker. From there it determines the shortest path from either the left or right key. The program will run in O() time because for every key it has to run left and right and check every locker on its way. There for each key gets run twice and each locker in its segment will get run for each time that segment is ran.

**Solutions:**

* **Set #1:** 99
* **Set #2:** 22
* **Set #3:** 68
* **Set #4:** 31
* **Set #5:** 103
* **Set #6:** 30
* **Set #7:** 87
* **Set #8:** 82