**Hasanain\_Alsabonchi**

**1. Consider the searching problem:**

**Input: A sequence of n numbers A = { a1, a2,...an} and a value v.**

**output: An index i such that v = A[i] or the special value NIL if v does not appear in A.**

**• Write a pseudocode for linear search, which scan through the sequence, looking for v.**

**• Using a loop invariant, prove that your algorithm is correct.**

**linear-search(A)**

**for i=1 to A.length**

**if A[j]==v then**

**return j**

**end if**

**end for**

**return nil**

**Prove it using loop variant**

**Initialization: we have an array A[1…….i-1] zero element or empty subarrayry and v is not in the list.**

**Maintenance: After execution the loop, A[i………..i-1] dose not have the key. even before start the loop**

**//is still false, unless v=A[i].**

**else A[1…ii] dose not contain the value**

**Termination:1)A.length+1, therefore it return nil**

**2) if A[i]= value return i ( the index of v in the list)**

2. Flipping pancakes:

There are n pancakes all of different sizes that are stacked on top of each other. You are allowed to slip a flipper under one of the pancakes and flip over the whole sack above the flipper. The purpose is to arrange pancakes according to their size with the biggest at the bottom. Design an algorithm for solving this puzzle.

Find the Biggest one put it on the top and flip over then find next biggest one and put second one from the bottom.

for i=1 to A.length

len=A.length-1

MAxindex=max(A,startIndex,EndInex) //find the max function

flip(arr,index) // 1 st flip from 0…………… maxIndex

flip(arr,len) // second flip reverse the whole array

max(A,s,e);

Maxindex=A[0]

for i=s to e

max=A[1]

if(A[i]>max)

max=A[i]

return Maxindex

swap(MaxIndex,A);

start = 0

**while** (start < i) :

            temp = arr[start];

            arr[start] = arr[MaxIndex];

            arr[MaxIndex] = temp;

            start++;

            MaxIndex--;

**3. Consider the following algorithm:**

**SelectionSort(A)**

**• for j = 1 to A.length - 1**

**•     min = A[j]**

**•     minIndex = j**

**•     for i = j + 1 to A.length**

**•         if A[i] < min**

**•              min = A[i]**

**•              minIndex = i**

**•    A[minIndex] = A[j]**

**•    A[j] = min**

3.1 Run the algorithm on A = [ 17, 4, 12, 25,  16, 10, 19]

3.2 Prove SelectionSort using loop invariant:

3.3 What loop invariant condition should we maintain?

3.4 Why do we run the outer loop only n - 1 times instead of n times?

**3.1**

MIN =17, MININDEX=1

**17, 4, 12, 25,  16, 10, 19 i=2 , j=1**

**4, 17, 12, 25,  16, 10, 19 min=4 ,minIndex=4**

**4, 17, 12, 25,  16, 10, 19 i=3 , j=2**

**4, 10, 12, 25,  16, 17, 19 min=10 ,minIndex=6**

**4, 10, 12, 25,  16, 17, 19 i=4 , j=3**

**4, 10, 12, 25,  16, 17, 19 i=5 , j=4**

**4, 10, 12, 25,  16, 17, 19 min=16 ,minIndex=5**

**4,10,12,16,25,17,19 i=6 , j=5**

**4,10,12,16,17,25,19 min=17 ,minIndex=6**

**4,10,12,16,17,25,19 i=7 , j=6**

**4,10,12,16,17,19,25 min=19 ,minIndex=6**

N-1 THIS WILL BW RUNNING

**3.2**

BEFORE WE RUN, HOW MNAY ELENT TO HAVE IN A A[1……0]=[] THUS A’ IS SORTED

**Initialization:**

we have a sorted list A[1………i-1]is sorted

**Maintenance:**

**MAINTAIN THIS LIST TO BE SORTED**

Before executing the inner loop, we know that A[1……i-1] is sorted.

After executing the inner loop,The Jth smallest element will be at A[j] , thus A[1….j] is sorted.

**or**

After executing the first loop ,we assume that the first element is the min element,After execution the second loop only one element will be added to the subarray A[1……j] “sub array is the sorted array”.

**Termination:**

When the algorithm Terminate j= A.length therefore the list will be sorted A[1……….n-1] will contain sorted list, where each A[i] contain i th smallest element in the list.

Thus A[j] contain the Largest Elelemnt, thus A is sorted

**3.3**

we should have sorted sub listA[1…….j] each time

**3.4**

because we comparing one element from subarray with other(next element) from the original array.

(The inner loop associate with outer loop )if we run the outer loop n times the inner loop will be out of index