Algorithm Assignment1

Xinyu Jiang

September 2018

1

Proof by induction:
Base case: when n=1, then 1=$\frac{1*(1+1)*(2*1+1)}{6} = 1$ which is true
Induction step: For all n $\in \mathbb{Z}+$ Assuming the statement is true for n=k, then $1^2 + 2^2 + 3^2 + ... + k^2 = \frac{k(k+1)(2k+1)}{6}$, then for n=k+1 the statement become $1^2 + 2^2 + 3^2 + ... + k^2 + (k+1)^2 = \frac{(k+1)(k+1+1)(2(k+1)+1)}{6}$,
simplify the right side, right side=$\frac{(k+1)(k+2)(2k+3)}{6}$
simplify the left side, from the statement above,
$(1^2+2^2+3^2+...+k^2)+(k+1)^2=\frac{k(k+1)(2k+1)}{6}+(k+1)^2=\frac{k(k+1)(2k+1)+6(k+1)^2}{6}=\frac{(k+1)(k(2k+1))+6(k+1)}{6}$ $=\frac{(k+1)(2k^2+7k+6)}{6}=\frac{(k+1)(k+2)(2k+3)}{6}$ which is true when n=k+1.
Conclusion: Since P(n+1)is correct, therefore by mathematical induction, P(n) is also correct, thus the statement above is correct

2

a)

```
linearCombining(A)
  for i=0 to A.length()-1,i++
    if($A[i] > v$)
    return i
  return NIL
```

b)
Loop invariant: A[0] to A[i-1] all element in the array have been checked it is smaller than the target value v.
Initialization: Showing that the loop invariant holds before the first iteration, when i=1,the subarray consists A[1,...i-1] consist with no element which holds the loop invariant
Maintenance:Showing that each iteration holds the loop invariant, for the loop A[1,...i-1] do not greater than v and A[i] does not greater than v, then it holds for the loop invariant
Termination: The for loop terminates if i=A.length-1 which means it has go

through all the element of the array and does not found any elements greater than v.

3

a)
If the array only contain one element, then the algorithm will subtract itself which will return 0.
If all element in the array are the same, then it will subtract itself all the time which will return 0
If the array do not have any element in there, which means it won't go into the for loop so it will return 0.
b)

```
MinPSoFar(A,length)
  if(length==1)
    return A[0]
  min=minPSoFar(A,length-1)
  if(A[length-1]¡min)
     return A[length-1]
  else
    return min

maxPF(A,minP,maxP,i)
//i starts from 0,maxP initialize to 0 in main function
  minP=minPSoFar(A,length of A+1)
  currMaxP=A[i]-minP
  i++
  if(i>length of A-1)
    return maxP
  else if(currMaxP>maxP)
    maxP=currMaxP
  return maxPF(A,minP,maxP,i)
```

c)
the implementation is in the zip file

4

a)

```
shrinkArr(A)
  if(elements in A<size of A/2)
    for(i to size of A/2)
      tempArr[i]=A[i]
    return tempArr
  else
```

```
    return A
```

//if elements in A is more than half of A, then we can not shrink it, just return the original array

b)

Loop Invariant:At the start of each iteration of the for loop, the new array tempArr[1,...i-1] which is half size of the old array contain all the elements from the old array A[1,...i-1]

c)initialization: Showing that the loop invariant hold before the for loop started, when i=1, the new array tempArr is empty and the size is half of the origional array which holds the loop invariant

Maintenance: the body of the for loop moves from A[1,...,i-1],for each iteration, there is element copy from array A to tempArr, therefore the new array tempArr contains all element from A which holds the loop invariant

Terminate: the loop goes through all the elements in A and copy all elements in A to new array which is half size of A, so it holds the loop invariant