Análisis de Algoritmos
Tarea 03: Justificación de algoritmos

Profesora: María de Luz Gasca Soto
Ayudantes: Rodrigo Fernando Velázquez Cruz
Teresa Becerril Torres
Nombre: Alvaro Ramirez Lopez
N°. de Cuenta.: 316276355
Correo: alvaro@ciencias.unam.mx

1. Problema 1

Considerar los siguientes problemas:

- Problema α: Calcular el producto de los elementos en el arreglo de números enteros $A[a..b]$.
- Problema β: Calcular la suma de los primeros n múltiplos de 3.
- Problema γ: Calcular la suma de los números impares en el arreglo de números enteros $A[a..b]$.

Elegir **dos** de los problemas anteriores, α o β o γ y ...

a) Proporcionar un algoritmo recursivo (código) que solucione el problema, indicando PreCondiciones y PostCondiciones

b) Demostrar que el algoritmo propuesto es correcto usando inducción matemática.

c) Calcular el tiempo de ejecución del algoritmo dado.

Solución:

2. Problema 2

Considera los siguientes problemas:

a) Problema Evaluación de Polinomios.

El Algoritmo Horner evalúa, en el punto x = xo, el polinomio

$P_n(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$

b) Problema Búsqueda Binaria. Dado un arreglo de enteros $A[a..b]$, ordenado en forma ascendente, determinar si el elemento x ∈ A e indicar la posición donde se encuentra, si está.

Dados los códigos para los problemas de A y B, mostrados abajo, demostrar usando la Técnica del Invariante del ciclo, que los algoritmos (códigos) son correctos.

```
Procedure H (A:Array of integers ; x0: Integer)
  var i, total : Integer;
    total ← 0 ;
      for i = n - 1 to 0 by -1 do
```

```
        tal ← A[i] + total * x0;
      Return total
end Horner


BinarySearch(var A:Atype; a,b:integer;
                 x:keytype; var pos:integer):boolean
// PreCondicion: a<=b+1 and A[a]<=...<=A[b]
var i,j,mid:integer; found: boolean;
begin
  i=a; j=b; found=false;
  while ( (i!=j+1) and (not found) ) do
    mid=(i+j) div 2;
    if (x = A[mid]) then found=true;
      elseif (x < A[mid]) then j=mid-1;
        else                   i=mid+1;
    end_if;
  end_while;
  if found then pos=mid;
    else pos=j;
  end_if;
  return found;
end BinarySearch

// PostC: (found implica: a<=pos<=b and A[pos] = x) and
          (not found implica: a-1 <= pos <= b
                     and (for all k a<=k<=pos, A[k]<x)
                     and (for all f pos+1<=k<=b, x<A[k]))
```

Solución:

Aquí va la solución.

3. Problema Opcional.

Para los problemas elegidos en el Ejercicio I

a) Proporcionar un algoritmo iterativo (código) que solucione el problema, indicando PreCondiciones y PostCondiciones

b) Demuestra que el algoritmo es correcto usando la técnica del Invariante del ciclo (loop invariant).

c) Determinar el desempeño computacional del algoritmo dado.

Solución:

Aquí va la solución.