Tarea 2

Nombre: Juan Sebastian Vargas
Fecha: 6/04/16

Punto 1

1.a

[Ctx C: A:[0..n-1]:**int** $\wedge$ $(\forall k | 0 \leq k < n$:A[k]=$a_k) \wedge$ b: int $\wedge n \geq 0$
i,z:=n-2,A[i+1]
{ Inv P: $0 \leq$ i < n $\wedge$ $z * b^i = (+k | i \leq k < n : a_k * b^k)$ }
{cota t: i }
do
i < n-1 $\wedge$ i $\geq 0 \rightarrow$ z=A[i]+$z * b$;i=i-1
od
{R: z = P.b }
]

1.b

la técnica que se uso para obtener la forma de P seria la de Apretar el cerco, conforme i cambia, P también, y con esto el producto $z * b^i$ también cambia, el intervalo en el que varia k se hace mayor, hasta que al final tenemos la definición del polinomio

1.c

el invariante siempre va a ser valido, para el caso inicial, cuando i=n-2 tenemos que; $zb^{n-2} = \sum_{k=n-2}^{n-1} a_k b^k = a_{n-2}b^{n-2} + a_{n-1}b^{n-1}$, pero z se puede escribir inductivamente $z = a_{n-2} + a_{n-1}b$, y de esta manera tenemos que: $zb^{n-2} = (a_{n-2} + a_{n-1}b)b^{n-2} = a_{n-2}b^{n-2} + a_{n-1}b^{n-1}$, facilmente demostrable por induccion vemos como el invariante aplica siempre.

1.d

si definimos la multiplicacion como operacion basica obtenemos que la complejidad temporal es de O(n), ya que en cada paso del DO se debe hacer una multiplicacion y suma, operaciones basicas, que se repiten n veces

Punto 2

2.a

{ Inv P: $0 \leq$ i < n }
{cota t: i }

2.b

Se utilizo la tecnica de Apretar el Cerco, ya que este problema se puede ver como un conjunto de valores que hace falta con calcular, que en cada iteracion se disminuye el numero de valores que falta por calcular, de este mismo modo y logica se escogio la cota, un numero que va disminuyendo y asi habra terminacion

2.c

[Ctx: N:nat $\wedge N > 0 \wedge$ f,h:[0..N-1]:**int** $\wedge$ f=F
i,h[0]:=1,f[0]
{ Inv P: 0 $\leq$ i < n }
{cota t: i }
do
i $\neq$ N $\rightarrow$ h[i]=f[i-1]-h[i-1];i=i+1
od
{ Pos R: $f[k] = \sum_{k=0}^{N-1} h[i]$}
]

2.d

Para la complejidad espacial tenemos dos arreglos de tamaños N, pero el orden solo seria de N, para la complejidad temporal tenemos un ciclo, y en cada pasada del ciclo se hacen dos sumas y es por esto que la complejidad seria de O($N * 2 * Tiempo(suma)$)=O($N * T(suma)$) si suponemos que T(suma)=1, esto es, el tiempo de una suma, obtenemos O(N)

Punto 3

Lenguaje se encuentra en el enunciado
Recurrencia:
pmax(1)=1
pmax(i+1)= max{ pmax(i),ppmaxf(i+1)}
ppmaxf(1)=1
ppmaxf(i+1)=1 si b[i]$\leq$ 0
ppmaxf(i+1) si b[i]>0
Estructura de datos
Se requiere solo guardar p, q, r, l
pro(r,s) : "b[r..r+s-1] es subarreglo", $0 \leq r \leq r + s - 1 \leq n$ **Codigo**
i,p,q:=1,1,1
r,l:=0,1
{ Inv P: 1 $\leq$ i $\leq$ n $\wedge p = ppmax(i) \wedge q = ppmaxf(i) \wedge pro(r, l)$}
do
i $\neq$ n $\rightarrow$ i=i+1
if b[i]> 0 then q:=qb[i];l=l+1 else q:=1;l=1 fi;
if $q > p$ then p=q;r:=i-l-1

od
{ Pos: p=ppmax(n) $\wedge$ pro(r,l) }