Complejidad de algoritmos

Taller 1 c++

Katherine Camacho - Juan José Hoyos

> *PUNTO 1*

int exponenciacion (int base, int exponente){

/*/

Esta funcion tiene por objetivo calcular la potencia de un número entero

cuya base y exponente se piden por teclado al usuario

```
/*/
int resultado = 1;  _____ 1

for (int i = 1; i <= exponente; i++){ ______ n+1-1
    resultado *= base;_____ n-1
}

return resultado;_____1
```

}

> ***Complejidad**=O(n)*

int busquedaSecuencial (int * A, int n, int k){

//

Esta funcion tiene por objetivo encontrar un elemento k ingresado por el usuario, el cuál

se buscará en un arreglo previamente definido, en caso de encontrarlo retorna la posición en

la cual está el elemento k, y si no lo encuentra retorna -1

//

int i=0;

```
while (i < n){_____n+1
    if ( A [i] == k){_____n
        return i;_____0
    }
    i++;_____n
}

return -1;_____1
```

}

> ***Complejidad**=n+1+n=2n*
>
> *O(n)*

void ordenamientoBurbuja (int * A, int n){

//

Esta funcion tiene por objetivo ordenar ascendentemente un arreglo con números enteros

recorriendo el arreglo con el fin de ir comparando cada uno de los datos dejando siempre el mayor de ultimo

para poder lograr organizarlo de la manera indicada.

//

int temp;

```
for (int i=0; i<=n-2; i++){____n-1
    for (int j=0; j<=n-2-i; j++){____((n(n+1))/2)+n-1
        if (A[j+1] < A[j]){_____((n(n+1))/2)
            temp = A[j];_______((n(n+1))/2)
            A[j] = A[j+1];______((n(n+1))/2)
            A[j+1] = temp;______((n(n+1))/2)
        }
    }
}
```

> ***Complejidad***=$O(n^2)$

void ordenamientoSeleccion (int * A, int n){

```
/*/
Esta funcion tiene por objetivo ordenar ascendentemente u
n arreglo con números enteros
recorriendolo con el fin de ir comparando cada uno de los
```

```
 datos para dejar siempre el valor
mas pequeño de primero, logrando así organizarlo de la ma
nera adecuada.
/*/
int temp;
int min;

for (int i=0; i<n-1; i++){ _____n
    min=i;_____n-1
    for (int j=i+1; j<n; j++){ _______((n(n-1))/2)+n-1
        if (A[j]< A[min]){__________((n(n-1))/2)
            min = j;______((n(n-1))/2)


        }
        temp = A[i];________((n(n-1))/2)
        A[i] = A[min];________((n(n-1))/2)
        A[min] = temp;________((n(n-1))/2)
    }
}
```

}

> ***Complejidad***$=O(n^2)$

int emparejamientoCadenas (char * T, int n, char * P, int m){

/*/

Esta funcion tiene por objetivo buscar la posicion de un patron ya

definido en una cadena que tambien ha sido
definida y es más grande, retornando siempre el valor que representa
la posicion de la primera letra del patron
a buscar.

```
/*/
int j;____1
for (int i=0; i<=n-m; i++){______n-m
    j=0;_____n-m-1
    while (j<m && P[j] == T [i + j]){_______(n-m-1)+m
        j += 1;___m
    }
    if (j == m){_____n-m-2
        return i; ____1
    }
}
return -1;_____ninguna
```

> ***Complejidad***=*O(n-m)*