Nombre: jose david gutierrez uribe.
Tecnicas y practicas de programacion.
Grupo A.

Análisis de algoritmos.

```
int pot(int a, int n) {
        int total=1;
        int z;
        for (int i = 1; i < n; i++) {                    n
                z = i;                                   n-1(1)
                total *=a;
        }
        cout << z + 1 << endl;
        return total;
}
```

la complejidad del algoritmo es:
O(n)
θ(n)
Ω(1)

```
int busquedaSec(int arreglo[], int n, int k) {
        int x = 0;
        while (x < n) {                                          n+1
                if (arreglo[x] == k) {
                        cout << "resul"<< x << endl;
                        return x;
                }
                x++;
        }
        return -1;
}
```

la complejidad del algoritmo es:
O(n)
θ(n/2) porque si el dato está en la mitad del arreglo solo se ejecutaría n/2.
Ω(1) sería una constante ya que entraría a la primera vez en el return.

```
void burbuja(int arreglo[],int sizer) {                  ejecuciones

        for (int i = 0; i < (sizer-2); i++) {            (n-1)

                for (int j = 0; j < (sizer - 2 - i); j++) {      ((n-2)(n-1)) / 2

                        if (arreglo[j + 1] < arreglo[j]) {
```

```
                    swap(arreglo, i, j);
                }
            }
        }

}
```

la complejidad del algoritmo es:
la complejidad es la misma en todos los casos ya que si o si se van a ejecutar n veces los for, sin importar que el arreglo está organizado.
O(n^2)
θ(n^2)
Ω(n^2)

```
void ordenSeleccion(int arreglo[], int sizer) {                    ejecuciones
        int min;        5-1=4                                      1
        0 1 2 3 4
        for (int i = 0; i < (sizer - 1); i++) {                    n
                min = i;                                           (n-1)(1)
                for (int j = i + 1; j < sizer ; j++) {             ((n-1)(n)) / 2

                        if (arreglo[j] < arreglo[min]) {
                                min = j;
                        }
                swap(arreglo, i, min);

                }
        }

}
```

la complejidad del algoritmo es:
selec sort y bubble sort son algoritmos muy parecidos y con orden por selección sucede lo mismo que en bubble se ejecutan si o si los for n vece.
O(n^2)
θ(n^2)
Ω(n^2)

```
int emparejamientoCadenas(string T[], string P[], int n, int m) {    ejecuciones
        for (int i = 0; i < (n - m); i++) {                           (n+1-m)
                int j = 0;

                while (j < m && P[j] == T[i + j]) {                   (n-m)(m+1)
                        j = j + 1;
                }
```

```
                if (j == m) {
                        return i;
                }
        }

        return -1;

}
```

la complejidad del algoritmo es:
O(n^2)
θ(n^2)
Ω(n^2)

En todos los caso se tiene que ejecutar los for ya que el algoritmo recorre los arreglos y compara si las letras si coinciden entonces esto lo obliga a que entre en los loops n veces.