Estructura de datos y de la información
Boletín de problemas - Tema 3

1. Calcular la función de coste de la siguiente secuencia tomando como operación elemental, primero, la comparación de elementos de vectores y, después, la asignación a elementos de vectores:

```
....
k=0;
for(i=0;i<(n-2);i++){
    for(j=i+1;j<n;j++){
        if(a[i]<b[j])
            c[k]=a[i];
        else
            c[k]=b[j];
        k=k+1;
    }
};
.....
```

2. Calcular la función de coste en multiplicaciones del siguiente fragmento de algoritmo:

```
....
/* Precondición: entero n, entero k y  n, k >= 1 */
i=1;
while(i<n){
    j=i-1;
    while(j<k){
        a[i]=a[i-1]+b[j+1]*b[j+2];
        j++;
    };
    i++;
};
....
```

3. Calcular, atendiendo al número de sumas, el orden de la función de coste del algoritmo:

```
int Sumas(int n){
    int i,j,s;

    i=0;
```

```
    s=0;
    while(i<=n){
        s=s+i;
        i=i+1;
        j=i;
        while(j<(2*(i-1))){
              s=s+j;
              j=j+1;
        }
    };

    return s;
}
```

4. Dos alumnos de Informática han quedado para estudiar programación y discuten por culpa del siguiente enunciado, *"Calcular el orden de coste del siguiente algoritmo tomando como operación elemental la comparación:"*

```
int Costoso(int a, int b, int N){
    int i,j,c;

    for(i=0;i<N;i++){
        j=0;
        while(j<i){
             if(j>=i)
                  j++;
             else
                  j=i;
        }
    };
    c=j;

    return c;
};
```

Uno dice que el coste en comparaciones es $O(N^2)$ en el peor caso y $O(N)$ en el mejor y el otro dice que no, que es $O(3N)$ tanto en el mejor como en el peor caso. ¿Quién tiene razón y por qué?

5. Calcular la función de coste del siguiente fragmento, en el mejor y en el peor caso:

```
....
/* Precondición: entero n >= 1, entero m >= 2 */
for(i=0;i<n;i++){
     j=m;
     while((j>2) || (i>n)){
          /* 2 operaciones elementales */
          j--;
     }
};
.....
```

6. Calcular la función de coste en el mejor y en el peor de los casos en el siguiente fragmento de algoritmo, tomando como operación elemental la multiplicación:

```
for(i=1;i<n;i++){
    j=0;
    while(j<n){
        if(a[i]<b[j]){
            j=j+1;
            c[j]=a[i]*b[j];
        }
        else{
            j=j+10;
            c[j]=a[j]*b[i];
        }
    }
};
```

7. Se tiene el siguiente fragmento de código:

```
for(i=0;i<n;i++){
    if (a>0)
        x[i]=x[i]+1;
    else
        x[i]=0;
};
```

   a) ¿Cuál es su coste en comparaciones?

   b) ¿Puedes escribir un algoritmo equivalente con un coste menor?¿Cuál es su coste?

8. Dado el fragmento de algoritmo:

```
for(i=0;i<n;i++){
    if (i==0)
        aux=v[i]*w[i];
    else
        aux=aux+v[i]*w[i];
};
```

   a) ¿Cuál es su coste en comparaciones?

   b) ¿Puedes escribir un algoritmo equivalente con un coste menor?¿Cuál es su coste?

9. Dados los algoritmos:

```
void Ejemplo(int a, int b, int N, int v[N]){
     int i,j;

     i=0;
     j=0;
     while(i<N){
         v[i]=a+j;
         j=Incremento(j);
         if(j>b)
             j=b;
         i=i+1;
     }
};

int Incremento(int x){
      int k, res;

      k=0;
      res=x;
      while(k<10){
          res=k*res;
          k=k+1;
      };

      return res;
};
```

Sabiendo que al analizar el algoritmo Ejemplo se ha obtenido como función de coste $13N + 1$ (valor exacto) ¿qué operación se ha considerado elemental en el análisis?

10. Dada la secuencia

```
for(i=0;i<n;i++){
    for(j=0;j<m;j++){
        if(A[i][j] > v[0])
            alg_cost_lineal(v,p,A[i][j]);
    }
};
```

y sabiendo que `alg_cost_lineal` tiene un coste de orden lineal, $O(p)$, ¿qué orden de coste tiene la secuencia anterior?

11. Calcular la función de coste en operaciones básicas del siguiente fragmento:

```
for(i=0;i<p;i++){
    j=0;
    while((m[i][j]!=0) && (j<q)){
        j++;
    };
    if(m[i][j]==0){
         Operacion_basica;
         for(k=j+1;k<q;k++){
              Operacion_basica;
         }
    }
};
```

12. Calcular la función de coste en operaciones básicas del siguiente fragmento:

```
i=0;
while((v[i]<c) && (i<n)){
    Operacion_basica;
    i++;
};
if(v[i]>=c){
    j=i;
    while(j<n)){
        Operacion_basica;
        j++;
    };
    j=0;
    while(j<=i){
```

```
        Operacion_basica;
        j++;
    }
};
```

13. Calcular el orden de coste de la siguiente secuencia, tomando la asignación como operación básica:

```
i=0;
s=0;
while(i<n){
    j=i;
    while(j<n){
        s=s+a[i][j];
        j=j+2;
    };
    j=i+1;
    while(j<n){
        s=s-a[i][j];
        j=j+2;
    };
    i=i+2;
};
```

14. Dada la secuencia

```
paso=1;
v[0]='0';
for(i=1; i<N; i++){
    for(j=0; j<paso; j++){
        if(v[j]=='1')
             v[paso+j]='0';
        else
             v[paso+j]='1';
    };
    paso=paso*2;
};
```

calcular su coste temporal en asignaciones. El resultado debe ser función de N **exclusivamente**. Nota:

$$\sum_{i=0}^{N-1}(2^i) = 2^N - 1$$

15. Diseñar un algoritmo para ordenar un grupo de N alumnos, para cada uno de los cuales se dispone de la siguiente información:

```
struct TAlumno  {
  int numexp;
  string nombre;
  char grupo;
};
```

de modo que, opcionalmente, se pueda obtener una relación ordenada de la siguiente forma:

- Opción (a) por orden alfabético,
- Opción (b) por número de expediente,
- Opción (c) por grupo, y dentro de cada grupo, por orden alfabético.

16. ¿Qué ventajas o inconvenientes presentaría esta formulación alternativa del algoritmo de selección, respecto de la estudiada?

```
void Selection2(vector v[], int N)
{   int i,j,min;
    int aux;

    for(i=0;i<N-1;i++){
        min=i;
        for(j=i+1;j<N;j++){
            if (v[j]<v[min])
                min=j;
        };
        if (min!=i){
            aux=v[i];
            v[i]=v[min];
            v[min]=aux;
        };
    };
 };
```

17. ¿Qué hace este algoritmo?¿Se ha estudiado alguno parecido en teoría? Relaciónalos y justifica qué posibles ventajas tendría esta nueva versión.

```
void Que_sera(int v[], int N)
{       int v[N]={ ... };
        int  k,aux, j, liminf, limsup, newliminf, newlimsup;

        liminf=0;
        limsup=N-1;
        while(limsup>liminf){
                newlimsup=liminf;
                for(j=liminf+1; j<limsup+1; j++){
                        if (v[j-1]>v[j]){
                                newlimsup=j;
                                aux=v[j-1];
                                v[j-1]=v[j];
                                v[j]=aux;
                         };
                };
                limsup=newlimsup;
                newliminf=limsup;
                for(j=limsup; j>liminf; j--){
                        if (v[j]<v[j-1]){
                                newliminf=j-1;
                                aux=v[j-1];
                                v[j-1]=v[j];
                                v[j]=aux;
                        };
                };
                liminf=newliminf;
         };
};
```

18. Alguien ha pensado que el algoritmo de ordenación por selección sería más rápido si en cada iteración se buscaran simultáneamente el máximo y el mínimo de la porción de vector que falta por odenar y se colocaran cada uno en su sitio. Escribir dicha versión alternativa y comparar el coste del algoritmo obtenido con el del original.

19. ¿Qué hace este algoritmo?

```
bool queHace(int v[], int i, int j, int x){

    bool enc;
    int m;

    if(i>j)
        enc=falso;
    else{
        m=(i+j)/2;
        if(v[m]==x)
               enc=cierto;
        else{
             if(v[m]>x)
                   enc=queHace(v,i,m-1,x);
             else
                   enc=queHace(v,m+1,j,x);
            }
        };

     return enc;
};
```

20. Dados los tipos

```
struct ganador {
       char[30] nombre;
       int puntos;
};

typedef ganador vector[N];
```

escribir un algoritmo que haga lo siguiente: Se sabe que `snood` está definido como de tipo `vector` y que está ordenado por orden decreciente por el valor del campo entero `puntos`. También se sabe que no todas las posiciones están ocupadas, sólo las `p` primeras (las posiciones libres tienen el campo `puntos` con el valor `-1`). Escribir un algoritmo al que se le pase el nombre de un jugador y su correspondiente puntuación y la inserte en su lugar correspondiente, de forma que el vector siga ordenado.