Asignatura: Fundamentos de algoritmia
Evaluación continua. Cuestionario.

Grupo A. Azul
Profesor: Isabel Pita.

Nombre del alumno:

1. Dadas las siguientes funciones de coste, indica su orden de complejidad más ajustado (representado por la función más sencilla) y ordena los órdenes de complejidad obtenidos utilizando $\subset$.

   a) $f_1(n) = 2^n + 3^n + 4^n + n^n$

   b) $f_2(n) = \frac{1}{10}n + \log^2 n$

   c) $f_3(n) = n\log_2^2 n + n\log_5 n + \log_{10} n$

   d) $f_4(n) = n\log n + n$

   e) $f_5(n) = \log_3 5n + \log_{10} 10n + \log_2 n^2$

   f) $f_6(n) = \log_3 10$

   Respuesta:

   a) $f_1(n) = 2^n + 3^n + 4^n + n^n \in \mathcal{O}(n^n)$.

   b) $f_2(n) = \frac{1}{10}n + \log^2 n \in \mathcal{O}(n)$.

   c) $f_3(n) = n\log_2^2 n + n\log_5 n + \log_{10} n \in \mathcal{O}(n\log^2 n)$.

   d) $f_4(n) = n\log n + n \in \mathcal{O}(n\log n)$.

   e) $f_5(n) = \log_3 5n + \log_{10} 10n + \log_2 n^2 \in \mathcal{O}(\log n)$.

   f) $f_6(n) = \log_3 10 \in \mathcal{O}(1)$.

   $\mathcal{O}(1) \subset \mathcal{O}(\log n) \subset \mathcal{O}(n) \subset \mathcal{O}(n\log n) \subset \mathcal{O}(n\log^2 n) \subset \mathcal{O}(n^n)$.

2. Dado el siguiente programa que elimina los valores negativos de un vector:

```
void f (std::vector<int> & v) {
  int i = 0;
  while ( i < v.size())
    if (v[i] < 0) {
        for (int j = i; j < v.size()-1; ++j) v[j] = v[j+1];
        v.pop_back();
    }
    else ++i;
}
```

   Indica su orden de complejidad justificando tu respuesta.

   Respuesta:

   El algoritmo es cuadrático en el número de elementos del vector.

   La función de coste es $\sum_{i=0}^{v.size()}(n-i) = n^2 - \sum_{i=0}^{v.size()} i = n^2 - \frac{n^2}{2} = \frac{n^2}{2} \in \mathcal{O}(n^2)$.

3. Dados los siguientes programas:

```
pair<int,int> f (vector<int> const & v) {
  int maxi = v[0], num = 1;
  for (int i = 1; i < v.size(); ++i)
    if (v[i] > maxi) {
      maxi = v[i];
      num = 1;
    }
    else ++num;
  return {maxi, num};
}
```

```
pair<int,int> f (vector<int> const & v) {
  int maxi = v[0];
  for (int i = 1; i < v.size(); ++i)
    if (v[i] > maxi) maxi = v[i];
  int num = 0;
  for (int i = 0; i < v.size(); ++i)
    if (v[i] == maxi) ++num;
  return {maxi, num};
}
```

a) Calcula la función de coste de cada uno de ellos, justificando brevemente la respuesta.

b) Indica su orden de complejidad.

c) Indica un caso peor para el primero de ellos.

Respuesta.

a) Función de coste (aproximada) del primer algoritmo: $5 * v.size() + 4$.

Justificación: La inicialización tiene dos asignaciones de coste constante más la inicialización del bucle `for` también de coste constante. El bucle da `v.size()` vueltas. En cada vuelta se realiza una comparación y un incremento en la cabecera de coste constante (obviamos la comparación extra cuando termina el bucle), y en el caso peor se ejecuta siempre la parte cierta del condicional que consta de una comparación y dos asignaciones todas de coste constante. El bucle por lo tanto realiza 5 operaciones de coste constante en cada vuelta. Por ultimo la instrucción `return` tiene coste constante y se realiza una sola vez.

Función de coste (aproximada) del segundo algoritmo: $4 * v.size() + 4 * v.size() + 5 = 8 * v.size() + 5$.

Justificación: La inicialización de las variables son dos asignaciones de coste constante mas la inicialización de las variables de control de los bucles, también de coste constante. La instrucción `return` también tiene coste constante. Hay dos bucles, el primero da `v.size()` vueltas, en el caso peor el coste de cada vuelta son dos comparaciones de coste constante (una en la cabecera y otra en el condicional) y dos asignaciones de coste constante (el incremento de la cabecera y la asignación del condicional. El segundo bucle da también `v.size()` vueltas y en cada vuelta hace dos comparaciones y dos incrementos todos de coste constante en el caso peor.

b) Los dos algoritmos están en $\mathcal{O}(n)$.

c) El caso peor ocurre cuando el vector está ordenado en orden creciente y se modifica el máximo en cada vuelta del vector.

4. Dada la siguiente expresión

$$\#k : 0 < k < v.size() : v[k] \neq v[k-1].$$

Indica que valor toma sobre el vector 5 2 2 4 7 7 5. Justifica tu respuesta.

Respuesta:

La expresión cuenta el número de elementos de un vector que son diferentes de la componente siguiente. El valor sobre el vector indicado es 4.

5. Escribe un predicado $diferentes(v)$ que exprese que todas las componentes de un vector v son diferentes.

Respuesta:

Existen muchas formas de expresarlo, algunas son:

- $\forall k1, k2 : 0 \leq k1 < k2 < v.size() : v[k1] \neq v[k2]$.
- $\forall k1, k2 : 0 \leq k1 < v.size() \wedge 0 <= k2 < v.size() \wedge k1 \neq k2 : v[k1] \neq v[k2])$.
- $\forall k : 0 <= k < v.size() : ((\#x : 0 <= x < v.size() : v[k] == v[x]) == 1)$.

6. Escribe un predicado $pot(x)$ que exprese que un número x es potencia de dos.

   Respuesta:

   - $\exists k : k \geq 0 : x == 2^k$

7. Utiliza el predicado anterior para expresar que todas las componentes de las posiciones pares de un vector v son potencia de dos.

   Respuesta:

   - $\forall k : 0 \leq k < v.size() \wedge k\,\%2 == 0 : pot(v[k])$

8. Indica las partes de una especificación *pre/post* y lo que se expresa en cada una de ellas. Propón un problema y escribe su especificación.

   Respuesta:

   - La *precondición* indica las propiedades que cumplen los datos de entrada.
   - La cabecera de la función indica los valores de entrada y salida.
   - La *postcondición* indica lo que cumplen los datos de salida.

   Ejemplo de un problema: Buscar un elemento en un vector ordenado

   P : $\{ \ \forall k : 0 \leq k < v.size() - 1 : v[k] < v[k+1] \ \}$

   buscar (vector<int> v, int e) dev bool b

   Q: $\{ \ b \equiv \exists k : 0 \leq k < v.size() : v[k] == e \ \}$