*Algoritmos, Datos y Programas – Preguntas de finales*

1. **Eficiencia**
2. **Defina el concepto de eficiencia**

Métrica de calidad de los algoritmos, asociada con una utilización optima de los recursos del sistema de cómputo donde se ejecutara el programa.

1. **Explique de que maneras puede medir la eficiencia de un algoritmo en tiempo y memoria**

Para medir la eficiencia de un algoritmo es fundamental contar con una medida del trabajo que realiza. Esta permitirá comparar los algoritmos y seleccionar la mejor implementación. Se analiza cada algoritmo en base al número de comparaciones y asignaciones.

1. **Especifique los métodos para medir el tiempo de ejecución de un algoritmo**

Existen dos formas:

* Realizar un análisis teórico: busca obtener una medida del trabajo realizado a fin de obtener una estimación de su tiempo de ejecución. Se calculan el número de comparaciones y de asignaciones que requiere el algoritmo. Análisis similares realizados sobre diferentes soluciones permiten estimar cual es la solución más eficiente.
* Realizar un análisis empírico: se basa en la aplicación de juegos de datos diferentes a una implementación del algoritmo, se manera de medir sus tiempos de respuesta. Tiene la ventaja de ser muy fácil de implementar, pero no tiene en cuenta la velocidad de la maquina, ni los datos con los que se ejecuta el algoritmo.

1. **¿Cuándo podemos afirmar que un programa es eficiente?**

Se puede afirmar que son eficientes aquellos programas que cumplen con la especificación del problema en el menor tiempo posible y aquellos que utilicen las estructuras de datos adecuadas de manera de minimizar la memoria ocupada.

1. **Corrección**
2. **Defina el concepto de corrección**

Un programa es correcto cuando cumple con la función especificada (requerimientos propuestos). Para determinar cuáles son esos requerimientos se debe tener una especificación completa, precisa y no ambigua del problema a resolver antes de escribir el programa. Para medir si un programa es correcto se debe probar con datos reales que permitan verificar su función.

1. **Técnicas para “demostrar” corrección. Descríbelas**

* Testing: su propósito es proveer evidencias convincentes de que el programa hace el trabajo esperado. Esta batería de pruebas (diseñadas adecuadamente) debe ser planteada y no implementada de forma aleatoria. Lo que cuenta es la calidad de las pruebas, no la cantidad. Si la prueba sale mal, el programa debe ser corregido y probado con la misma batería de test que las que tuvo cuando fallo.
* Debugging: es el proceso de descubrir un error en el código y repararlo. Los errores pueden aparecer en el programa de distintas formas:
  + Problemas en el diseño del programa, será necesario corregirlo o rehacerlo.
  + La utilización de un proceso erróneo.
  + Codificación incorrecta.
  + Errores sintácticos, semánticos o lógicos.
  + Errores de hardware o errores de sistema (poco probables).
  + El programa puede ser aplicado a situaciones no previstas produciendo resultados incorrectos.

A veces hace falta poner breakpoints o checkpoints, o incluso hacer un seguimiento paso a paso (trace), inspeccionando los valores de las variables hasta encontrar el error.

* Walkthroughs: la idea es seguir o leer el código de programa ya escrito intentando comprender, o haciendo comprender a otro como funciona. Un programa bien estructurado es fácil de leer.
* Verificación: cuando probamos nuestro programa, observamos y analizamos su comportamiento. Sobre la base de este análisis podemos hacer ciertas afirmaciones del tipo: si nuestro programa es utilizado con ciertos datos, produce ciertos resultados. En otras palabras se verifica si se cumplen las pre y post condiciones del programa íntegramente.