aprenderaprogramar.com

Verificación ordenada y aleatoria de algoritmos. Uso de la potencia del ordenador. (CU00234A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel II

Fecha revisión: 2024

Autor: Mario R. Rancel

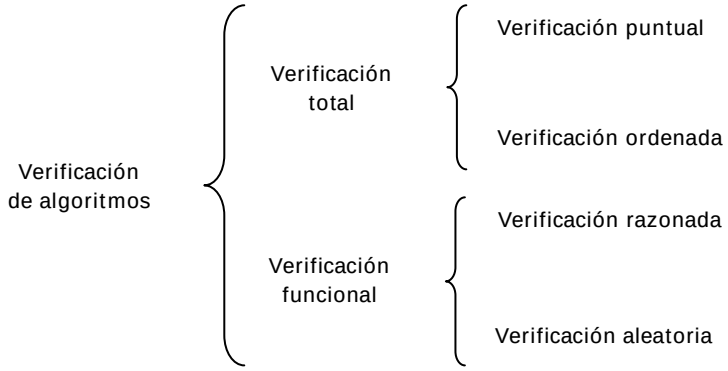
Resumen: Entrega nº 33 del Curso Bases de la programación Nivel II

24

ALGORITMOS GENÉRICOS. VERIFICACIÓN FUNCIONAL Y VERIFICACIÓN TOTAL

Dentro de las técnicas descritas para la verificación de algoritmos indicamos la posibilidad de uso del ordenador para realizar una "programación rápida". Esta técnica, para el desarrollo de algoritmos genéricos, puede resultar extraordinariamente útil y a ello vamos a dedicar las próximas líneas.

Con el fin de ordenar ideas vamos a ver una clasificación de vías de verificación de un algoritmo basada no en la técnica empleada, sino en el enfoque que el programador hace de la verificación.



Llamamos ***Verificación puntual*** a una prueba de algoritmo en que los datos de entrada son únicos de forma que, una vez realizada la verificación, la fiabilidad es total.

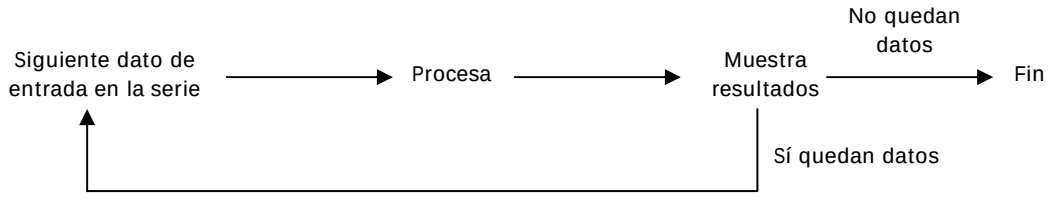
Verificación ordenada sería el conjunto de pruebas con una serie de datos de entrada finitos los cuales son probados uno por uno. Terminado el proceso se conoce que la fiabilidad es total.

Verificación razonada comprendería lo expuesto anteriormente en relación con la verificación de algoritmos genéricos: el número de posibles datos de entrada es muy amplio y no es viable una verificación total, por lo que se recurre a la elección de una serie de casos representativos. El programador valora qué casos elegir y por qué.

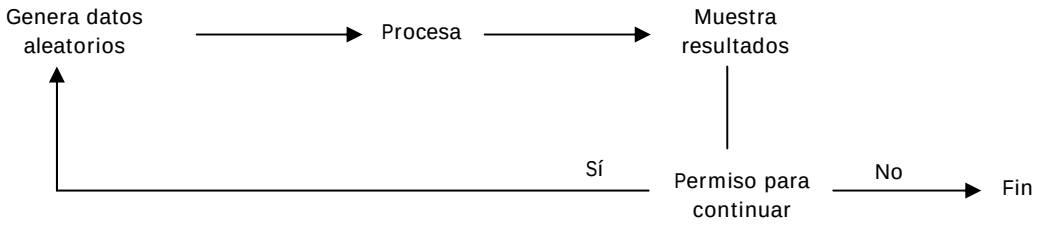
Por último, la ***verificación aleatoria*** partiría también de un amplio rango de datos de entrada que hacen no viable una verificación total, pero los casos que se prueban se escogen de forma aleatoria. El número de casos a probar puede estar definido de antemano por el programador o bien éste comienza la prueba y va observando resultados hasta, en un momento dado, darla por buena. Cualquiera de las formas de verificación expuestas puede ser desarrollada con alguna de las técnicas vistas, pero la verificación ordenada y la verificación aleatoria "invitan" al uso del ordenador. El número de casos a procesar suele ser relativamente elevado, por lo que el uso de tablas de variables puede requerir un tiempo excesivo.

Esquemáticamente los procesos serían estos:

Verificación ordenada



Verificación aleatoria



Los datos aleatorios se pueden preparar para que entren en paquetes, de forma que no hay que estar dando permiso para continuar uno a uno. Por ejemplo, que entren 10 datos habiendo 10 procesos y 10 resultados, tras cuya evaluación decidimos si generar otros 10 ó finalizar. La verificación aleatoria es muy útil cuando no se encuentran criterios para elegir casos o cuando queremos realizar un número de pruebas grande sobre un rango de datos amplio. El término "aleatorio" no debe confundirse con "sin ton ni son". Cuando se manda al ordenador a generar un número entero, por ejemplo, éste puede ser muy pequeño (supongamos 3) o muy grande (como 30000). Conviene pensar bien qué datos son aceptables y cuáles nos pueden causar problemas. Si por ejemplo en el bucle:

```
[Ejemplo aprenderaprogramar.com]
1. Desde i = 1 hasta b Hacer
        2. Desde j = 1 hasta n Hacer
                3. A = i * j
                4. Mostrar A
        5. Siguiente j
6. Siguiente i
```

Establecemos *b* y *n* como aleatorios enteros sin más, podría ocurrir que *b* tome valor 29683 y *n* valor 27324. Imaginemos la tabla de variables para estos numeritos: ¡descomunal! Ni siquiera la potencia del ordenador podrá con ellos y quedará bloqueado.

En estas situaciones se realiza una generación de aleatorios acotada entre un valor máximo y mínimo. En este ejemplo podríamos poner:

```
b ← aleatorio entre 2 y 20

n ← aleatorio entre 2 y 5
```

No hemos visto las instrucciones para generar números aleatorios, que no tienen mayor complicación. Las dejaremos para su análisis cuando hablemos de lenguajes concretos como pueden ser *Visual Basic, C, C++, Pascal, Java, etc*.

Próxima entrega: CU00235A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=36&Itemid=60