Especificación segundo proyecto

tema

Generando funciones

Versión 1.0

Valor 10 %

Preparado por Eddy Ramírez

Índice general

1 Introducción

Este documento contiene la especificación del segundo proyecto programado para el curso de Lenguajes de Programación, correspondiente al primer semestre 2014.

1.1. Propósito

El proyecto tiene como objetivo estudiar el paradigma funcional, en particular el trato de funciones numéricas, manipulación de listas y comunicación entre programas.

1.2. Alcance

Se espera que el estudiante desarrolle un sistema capaz de encontrar una solución satisfactoria a la creación de funciones basadas en lambda para diferentes distribuciones.

1.3. Referencias

1.

2 Descripción general

En muchos casos los programas de simulación necesitan generar números aleatorios que correspondan con una distribución de probabilidad dada, la cual puede ser discreta (Poisson, binomial, geométrica, hipergeométrica, etc) o continua (normal, exponencial, Ji-Cuadrada, Kolmogorov-Smirnoff, gamma, F, etc.). Incluso, en algunas ocasiones, puede tratarse de distribuciones que siguen comportamientos atípicos, por lo que no va a corresponder con ninguna de las distribuciones mencionadas anteriormente.

El problema consiste en construir un generador aleatorio que provea muestras con una distribución en particular y permita dibujar las frecuencias en que ocurren dichas muestras, para así corroborar el nivel de eficacia del programa con respecto de la gráfica de la función esperada.

2.0.1. Creación del laberinto

Se crearán una serie de funciones que indiquen la probabilidad de cada una de las siguientes distribuciones:

Distribuciones discretas:

- Tabla de probabilidad
- Poisson
- Binomial
- Geométrica
- Hipergeométrica
- Uniforme discreta

Distribuciones continuas:

- Normal
- Exponencial
- Uniforme continua

En el caso de las funciones continuas, por ejemplo, la normal estándar (con media 0 y desviación estándar 1), la probabilidad de que un x sea igual a la media, es cero, porque

en principio hay infinitos números entre cualquier intervalo [-x, x]. Sin embargo, las probabilidades en las funciones continuas están dadas por rangos, entonces si podemos calcular la probabilidad de que una variable esté dentro de un rango particular.

Para poder calcular el área bajo la curva de una función f en el intervalo [a, b] se debe poder calcular la integral de esa función en el rango deseado. A dicha área la llamaremos "A". Luego, para obtener la probabilidad de que haya una variable aleatoria en ese espacio, debe de calcularse la integral desde menos infinito hasta infinito. A dicha área la llamaremos "B". Finalmente para poder calcular la probabilidad de que un x esté en dicha área, basta con dividir A/B.

Vale la pena mencionar que el cálculo de la integral se puede realizar por medio del método de Simpson

2.1. Funcionamiento del programa

El programa debe de poder leer un archivo de texto (más adelante se explicará el formato del archivo), el cual tiene una distribución sobre la cual deben de generarse números aleatorios. Además debe de conectarse por socket con un programa hecho en Java que despliegue un gráfico de frecuencias con que se generan los valores en el programa de Scheme.

2.2. Nombres de las funciones, entradas y salidas

El programa recibe diferentes funciones de acuerdo con la siguiente tabla:

Función	***Nombre de la función***	***Entrada(s)***	***Salida***
Tabla de probabilidad	tabla	Una lista de listas con el siguiente formato: '$((x_1\ P_{x_1})(x_2\ P_{x_2})\ldots(x_n\ P_{x_n}))$ Donde: $\sum_{i=1}^{n} P_{x_i} = 1$ Por ejemplo: '((1 0.2) (2 0.4) (3 0.4))	Una función λ tal que dado un valor aleatorio [0,1[retorne un valor x_i que siga la distribución según la tabla.
Poisson	poisson	Un valor para su media (λ) Ejemplo: (poisson 3)	Una función λ que para un valor [0,1[retorne un x de poisson
Binomial	binomial	El tamaño n de la cantidad de experimentos y la probabilidad p de éxito Ejemplo: (binomial 100 0.3)	Una función λ que para un valor [0,1[retorne un x que siga la distribución binomial
Geométrica	geometrica	Una probabilidad de éxito p con 0 <= p <= 1. Ejemplo: (geometrica 0.3)	Una función λ que para un valor [0,1[retorne un x geométrico
Hipergeométrica	hipergeometrica	Recibe N,d y n, que son: N = tamaño de la población d = tamaño de los "exitosos" n = muestra seleccionada Ejemplo: (hipergeometrica 100 15 30)	Una función λ que dado un valor [0,1[retorne un valor x que siga una distribución hipergeométrica
Uniforme Discreta	uniforme-disc	Recibe una lista de valores todos con la misma probabilidad de ser escogidos Ejemplo: (uniforme-disc '(2 4 6 8))	Una función λ que retorne de manera uniforme alguno de los valores de la lista
Exponencial	exponencial	Recibe un valor para su μ ($\frac{1}{\lambda}$) es decir, para su media Ejemplo: (exponencial 3))	Una función λ que retorne para un valor [0,1[, un x exponencial
Normal	normal	Recibe dos valores, μ y φ, es decir la media y la desviación estándar. Ejemplo: (normal 0 1)	Una función λ que dado un valor [0,1[que retorne un valor x con una probabilidad normal
Uniforme Continua	uniforme	Recibe dos valores que son los rangos entre los cuales se generará los valores. Ejemplo: (uniforme 0 1)	Una función λ sin parámetros que retorna un valor continuo en el rango dado

2.3. Configuración de entrada

El programa recibirá un archivo de entrada que le indicará como debe de ejecutarse.

El archivo que se debe de leer tiene el siguiente formato: Línea 2: Cantidad de valores a generar Línea 4: Si la distribución es uniforme (que no necesita rango) o si es continua o discreta Línea 6: Indica el rango donde debe de calcular el valor de probabilidad y en caso de ser continua el tamaño del intervalo de la probabilidad. Línea 4 y sgtes: Definición en Scheme de algunas funciones en tiempo de ejecución.

Ejemplos:

Archivo 1
;; Cantidad de valores a generar
100000
;; Tipo de función
discreta
;; Rango
(0 15)
;; Definición de la función
(poisson 2)

Archivo 2
;; Cantidad de valores a generar
2000
;; Tipo de función
uniforme
;; Definición de la función dado
(uniforme-discreta '(1 2 3 4 5 6))

Archivo 3
;; Cantidad de valores a generar
500000
;; Tipo de función
uniforme
;; Definición de la nueva función
(uniforme 0 1)

Archivo 4
;; Cantidad de valores a generar
150000
;; Tipo función
continua
;; Rango
(0 1 0.1)
;; Definición de la nueva función

```
(lambda (x)
(cond ((< 0.5 ((uniforme 0 1) x)) (* 2 x))
(else (+ 1 x))))
```

2.4. Consideraciones de la implementación

El programa que genera los números aleatorios debe de ser en Scheme.

Es importante destacar que el uso de variables y la función set! quedan estrictamente prohibidos. La función let únicamente se puede utilizar para la función de socket y de leer archivo.

En el caso del graficador, el lenguaje que deben de utilizar será Java. En este caso no hay restricción en cuanto a los objetos o bibliotecas que deseen utilizar.

2.5. Aspectos generales

El proyecto se puede realizar en grupos de dos personas, como máximo.

Debe presentarse un avance del proyecto, para verificar su desarrollo. Este avance no será calificado, pero si no es presentado o es presentado incompleto, se rebajará hasta un 15

Incluya comentarios en el código de los programas y describa detalladamente cada una de las funciones utilizadas.

En caso de que la aplicación no funcione adecuadamente, efectúe un análisis de los resultados obtenidos, indicando las razones por las cuales el programa no trabaja correctamente, y cuáles son las posibles correcciones que se podrían hacer. De existir disponibilidad de un laboratorio, se revisaría el proyecto en clase, durante la semana siguiente a la entrega. Durante la revisión del proyecto, es muy importante poder defender adecuadamente la solución propuesta.

La documentación que debe tener este proyecto es muy sencilla y debe de tener lo siguiente:

1. Una descripción del problema con sus propias palabras
2. El método de solución que emplearon para resolver el problema
3. Un árbol de invocación de funciones (donde la raíz es la función principal y las hojas las más básicas), para este árbol no es necesario incluir las funciones auxiliares.
4. Una muestra de resultados experimentales que incluya capturas de pantalla del programa en ejecución.
5. Texto de los archivos utilizados en pruebas.
6. Un manual de usuario que explique como debe de ejecutarse el programa (tanto en Scheme y en Java)

2.5.1. Evaluación

Cada distribución discreta y la uniforme continua tienen un valor de 5 %. La normal y exponencial 7.5 % cada una.

El graficador en Java tiene un valor de 15 % y la comunicación por medio de sockets, 10 %.

2.5.2. Aspectos técnicos

1. Toda la programación debe de realizarse sobre GNU/Linux
2. El programa debe de tener un makefile

2.5.3. Aspectos administrativos

1. Se debe entregar sólo el código fuente y el makefile
2. Cualquier sospecha de fraude implica una nota de cero y se aplicarán las sanciones respectivas
3. La tarea se entrega a través de los correos: proyectos.eddy@gmail.com y proyectos.eddy.cartago@gmail.com para el 25 de abril.