UNIVERSIDAD DE GRANADA

Simulación de Sistemas

grado en ingeniería informática

Ejercicio de MonteCarlo

Fábrica de chocolate

Autor

José María Sánchez Guerrero

Rama

Computación y Sistemas Inteligentes

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice

1. Introducción

Una fábrica de chocolate recibe todos los años para diciembre un pedido de huevos de Pascua. Por razones estacionales, resulta más barato comprar el chocolate necesario durante el mes de **agosto**, asi que la empresa compra una gran cantidad de chocolate este mes, y si es necesario comprar más, se realiza otro **pedido adicional** para satisfacer de forma exacta toda la demanda. Por otro lado, si el chocolate comprado en agosto sobra, será donado a comedores de escuelas.

También tenemos los siguientes datos sobre precios y cantidades:

- Cada huevo de pascua emplea 250 gramos de chocolate.
- El precio del chocolate en agosto es de 1 euro por kilo.
- El precio del chocolate en diciembre es de 1.5 euros por kilo.
- El precio de venta de los huevos de pascua es de 0.60 euros la unidad.

La demanda de huevos al año sigue una distribucion triangular, con valor más probable es **c = 2600 unidades**, el menor valor es **a = 2000 unidades** y mayor valor es **b = 3000 unidades**. Su función de densidad es la siguiente:

$$f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & a \leq x \leq c \\ \frac{2(b-x)}{(b-a)(b-c)} & c \leq x \leq b \end{cases}$$

Con esta información tendremos que construir un modelo de simulación

2. Implementación del generador

La implementación de este generador triangular ya se nos proporcionará en el enunciado. Primero necesitamos un primer generador de números aleatorios:

```
float uniforme(){
  float u;
  u = (float) random();
  u = u/(float)(RAND_MAX+1.0);
  return u;
}
```

Y después lo utilizamos en la función de distribución comentada anteriormente:

```
int generademanda(){
  float u, x;
  u = uniforme();

  if (u < (c-a)/(b-a))
    x = a+sqrt((b-a)*(c-a)*u);
  else
    x = b-sqrt((b-a)*(b-c)*(1-u));

  return (int) x; // Se convierte a entero porque es la demanda
                  // de huevos de pascua
}
```

Se nos pide crear un modelo de simulación para este sistema que determine la cantidad de kilos de chocolate que tenemos que comprar en el mes de agosto para optimizar las ganancias de la empresa.

3. Implementación del modelo

En el apartado anterior ya hemos explicado cómo funciona el generador triangular de nuestro modelo, pero no hemos explicado dónde ni cómo lo utilizamos. Esto lo vamos a hacer en esta sección.

Lo primero es declarar e inicializar las variables que vamos a utilizar:

```

  // Variables globales //
  int a = 2000, b = 3000, c = 2600;

  // Variables locales //
  double x = 0.6,    // Ganancia por unidad vendida
       y1 = 0.25,    // Perdida por unidad no vendida con chocolate
                     // comprado en agosto
       y2 = 0.375;   // Perdida por unidad no vendida con chocolate
                     // comprado en diciembre

  int veces = 10000;
  int demanda, ganancia, s_maxima;
  double ganancia_maxima = 0;

```

Los valores que tienen las variables x, $y1$ y $y2$ vienen dados por las condiciones explicadas en la introducción. La x es el precio en euros obtenido por unidad vendida, $y1$ es el precio por unidad que no se vende si su chocolate ha sido comprado en agosto (250gr entre 1eu/kg); y $y2$ es el precio por unidad que no se vende si su chocolate ha sido comprado en diciembre (250gr entre 1.5eu/kg).

A continuación, para estimar la ganancia esperada, utilizamos el generador de datos a partir de la distribución triangular y repetimos ese proceso un número de veces determinado (y calcular al final un promedio de ellas). Lo más importante en un modelo de MonteCarlo es muestrear el sistema en estudio para estimar las características a partir de los datos de la muestra.

Como ya tenemos el generador implementado, el núcleo de nuestro proceso es el siguiente:

```
1
2   // Inicializamos contadores
3   double sum = 0.0, sum2 = 0.0;
4
5   for (int i = 0; i < veces; i++){
6     demanda = generademanda();
7
8     if (s > demanda)
9       ganancia = demanda*x - s*y1;
10
11    else
12      ganancia = demanda*x - (s*y1 + (demanda-s)*y2);
13
14    sum += ganancia;
15    sum2 += ganancia*ganancia;
16  }
17
18  // Obtener ganancia media y desviacion tipica
19  double ganancia_esperada = sum/veces,
20         desviacion = sqrt((sum2-veces*ganancia_esperada*
      ganancia_esperada)/(veces - 1));
21
22  // Comprobamos si es la ganancia maxima
23  if (ganancia_esperada > ganancia_maxima){
24    ganancia_maxima = ganancia_esperada;
25    s_maxima = s;
26  }
27
28  // Imprimimos resultado de cada iteraccion
29  printf("s: %d, ganancia: %f, desv: %f\n", s, ganancia_esperada,
      desviacion);
30
```

Repetimos este proceso para diferentes valores de s, los cuales están proporcionados en los datos del problema y establecidos como variables globales. Por último, generamos un informe final con un resumen de los datos de la ejecución y con los mejores datos obtenidos.

```
1   printf("\nValor de x: %f, valor de y1: %f, valor de y2: %f, numero
      de veces: %d", x, y1, y2, veces);
2   printf("\nValor maximo de ganancia: %f || s --> %d\n",
      ganancia_maxima, s_maxima);
```

4. Experimentación y análisis del modelo

Vamos a ejecutar nuestro modelo con un número alto de iteracciones, por ejemplo 10000, ya que su ejecución es bastante rápida. Cuantas más iteracciones pongamos, más fiel a la realidad será nuestro modelo, teniendo siempre en cuenta la aleatoriedad del generador. Estos son los resultados que hemos obtenido:

```
$ ./montecarlo
s: 2000, ganancia: 829.460400, desv: 33.825544
s: 2001, ganancia: 829.396200, desv: 33.503580
s: 2002, ganancia: 829.556500, desv: 33.546507
s: 2003, ganancia: 829.930000, desv: 33.295441
s: 2004, ganancia: 829.995900, desv: 33.459098

.............

s: 2996, ganancia: 795.998700, desv: 88.730445
s: 2997, ganancia: 795.683900, desv: 88.670787
s: 2998, ganancia: 798.665700, desv: 90.704331
s: 2999, ganancia: 798.103000, desv: 89.634769
s: 3000, ganancia: 796.020700, desv: 88.600780

Valor de x: 0.600000, valor de y1: 0.250000, valor de y2: 0.375000,
numero de veces: 10000
Valor maximo de ganancia: 883.158100 || s --> 2496

```

Como podemos ver, el resultado con el que hemos obtenido una mayor ganancia ha sido **2496 kg** de chocolate comprado en agosto. Este valor es bastante lógico por dos razones. La primera es que es un valor cercano al pico de la función triangular (que es b = 2600), y por otra parte, se nos ha quedado un poco por debajo de éste debido a que se penaliza más el tener que pagar un sobrante excesivo, a volver a comprar en diciembre hasta llegar a la cantidad justa de chocolate deseado.

Si ejecutamos varias veces más, vemos como los valores de ganancia máximo se mantienen en un rango cercano.

```

  Valor de x: 0.600000, valor de y1: 0.250000, valor de y2: 0.375000,
  numero de veces: 10000
  Valor maximo de ganancia: 883.370100 || s --> 2495

  Valor de x: 0.600000, valor de y1: 0.250000, valor de y2: 0.375000,
  numero de veces: 10000
  Valor maximo de ganancia: 883.164300 || s --> 2460

  Valor de x: 0.600000, valor de y1: 0.250000, valor de y2: 0.375000,
  numero de veces: 10000
  Valor maximo de ganancia: 883.170200 || s --> 2467
```

```
13
14  Valor de x: 0.600000, valor de y1: 0.250000, valor de y2: 0.375000,
15  numero de veces: 10000
16  Valor maximo de ganancia: 883.512100 || s --> 2498
17
18  Valor de x: 0.600000, valor de y1: 0.250000, valor de y2: 0.375000,
19  numero de veces: 10000
20  Valor maximo de ganancia: 883.213600 || s --> 2483
21
```

Como conclusión, si tuvisemos que elegir una cantidad de chocolate que comprar en verano yo me decantaría por un valor cercano a los **2480 kg**, teniendo en cuenta las dos razones comentadas anteriormente y que es un valor medio a las distintas ejecuciones del modelo que hemos realizado.