

Simulación de Intercambio de Dulces y Obtención de Paletas

Alumno: NEXU YOHAN MAMAMI YUCRA

Descripción del Problema

Se desea simular un sistema en el que participan **9 estudiantes**, los cuales reciben dulces y pueden canjearlos por paletas (chupetines) siguiendo ciertas reglas. El objetivo es que **cada estudiante termine con exactamente una paleta**, realizando el menor número posible de intercambios o movimientos.

Reglas del Sistema

- Existen tres tipos de dulces: **A, B y C**.
- En cada ronda, cada estudiante recibe **2 dulces aleatorios**.
- Si un estudiante reúne al menos:
 - Un conjunto de (**A, B, C**): puede canjearlo por una **paleta**.
 - Dos conjuntos (**2A, 2B, 2C**): puede canjearlos por una **paleta** y recibe un **dulce adicional al azar**.
- Un estudiante sólo puede tener **una paleta como máximo**.
- Existe una pequeña probabilidad (**15 %**) de que un estudiante decida **devolver su paleta**, recibiendo a cambio **3 dulces aleatorios**.
- El proceso continúa hasta que **los 9 estudiantes posean exactamente una paleta**.

Código en R

```
1 set.seed(123)
2
3 simular_paletas_unicas <- function() {
4   # Crear 9 estudiantes con contadores de dulces A, B, C y
      paletas P
5   estudiantes <- vector("list", 9)
6   for (i in 1:9) estudiantes[[i]] <- c(A=0, B=0, C=0, P=0)
7
8   iteraciones <- 0
```

```

9   contador_paleta <- 1
10
11  cat("INICIO DE LA SIMULACION DE DULCES Y PALETAS (1 paleta por
12     estudiante)\n")
13
14  while (TRUE) {
15     iteraciones <- iteraciones + 1
16     cat("\nIteracion", iteraciones, "\n")
17
18     # 1. Repartir 2 dulces aleatorios a cada estudiante
19     for (i in 1:9) {
20         nuevos <- sample(c("A", "B", "C"), 2, replace = TRUE)
21         for (candy in nuevos) estudiantes[[i]][candy] <-
22             estudiantes[[i]][candy] + 1
23     }
24
25     # 2. Canjes y devoluciones
26     for (i in 1:9) {
27         # Solo estudiantes sin paleta pueden intentar canjear
28         if (estudiantes[[i]]["P"] == 0) {
29             sets <- min(estudiantes[[i]][c("A", "B", "C")])
30
31             if (sets >= 2) {
32                 # Canje por 6 dulces (2 sets) -> 1 paleta + 1 dulce al
33                 # azar
34                 estudiantes[[i]]["P"] <- 1
35                 estudiantes[[i]][c("A", "B", "C")] <- estudiantes[[i]][c(
36                     "A", "B", "C")] - 2
37                 bonus <- sample(c("A", "B", "C"), 1)
38                 estudiantes[[i]][bonus] <- estudiantes[[i]][bonus] + 1
39                 cat(sprintf(" Paleta #%d -> Estudiante %d (2 sets + %s
40                     extra)\n", contador_paleta, i, bonus))
41                 contador_paleta <- contador_paleta + 1
42             } else if (sets == 1) {
43                 # Canje por 3 dulces (1 set) -> 1 paleta
44                 estudiantes[[i]]["P"] <- 1
45                 estudiantes[[i]][c("A", "B", "C")] <- estudiantes[[i]][c(
46                     "A", "B", "C")] - 1
47                 cat(sprintf(" Paleta #%d -> Estudiante %d (1 set)\n",
48                     contador_paleta, i))
49                 contador_paleta <- contador_paleta + 1
50             }
51         }
52     }
53
54     # 3. Regla de devolucion (solo si ya tiene paleta)
55     for (i in 1:9) {
56         if (estudiantes[[i]]["P"] == 1 && runif(1) < 0.15) {
57             estudiantes[[i]]["P"] <- 0
58             devolucion <- sample(c("A", "B", "C"), 3, replace = TRUE)

```

```

53     for (candy in devolucion) estudiantes[[i]][candy] <-
        estudiantes[[i]][candy] + 1
54     cat(sprintf("  Estudiante %d devolvio su paleta -> obtuvo
        dulces %s\n",
55                 i, paste(devolucion, collapse=" ")))
56   }
57 }
58
59 # 4. Condición de parada: todos con exactamente 1 paleta
60 paletas_por_est <- sapply(estudiantes, function(e) e["P"])
61 if (all(paletas_por_est == 1)) break
62 }
63
64 # 5. Resultado final
65 cat("\nIteraciones totales:", iteraciones, "\n")
66 tabla <- t(sapply(estudiantes, function(e) e))
67 colnames(tabla) <- c("A", "B", "C", "Paletas")
68 print(as.data.frame(tabla))
69 cat("\nTodos los estudiantes tienen exactamente UNA paleta\n")
70 }
71
72 # Ejecutar simulación
73 simular_paletas_unicas()

```

Conclusión

Este experimento computacional permite observar cómo, a través de un sistema de reglas de canje y aleatoriedad controlada, los jugadores (estudiantes) pueden alcanzar el equilibrio de tener una paleta cada uno con un número mínimo de intercambios. La simulación también sirve como ejemplo práctico de estrategias de optimización en sistemas cerrados con recursos limitados.