Actividad 8 - Variado

UNSAM Programa - Adaptado de Exactas Programa

Verano 2020

Hoy vamos a plantear 3 problemas muy conocidos y un acertijo de yapa.

1. Tirando dardos.

Vamos a realizar una adaptación del problema de la aguja, planteado originalmente por el Conde de Buffon en el siglo XVIII.
En un panel de corcho cuadrado colgamos un blanco de dardos, que tiene el mismo diámetro que el lado del cuadrado (ver figura 1.)

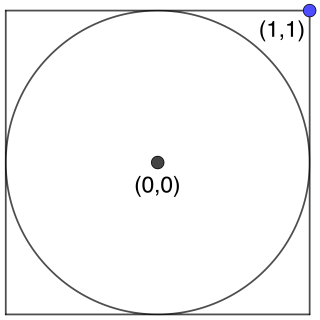

Figura 1: La plancha de corcho y el blanco de dardos

Tiramos varios dardos y suponemos que nunca caen fuera del cuadrado. Llevamos la cuenta de cuántos dardos tiramos en total (`n`) y de cuántos fueron al blanco (`blanco`) y guardamos el cociente `blanco/n`.
Para todo el problema vamos a suponer que el blanco es el círculo de centro con coordenadas (0,0) y que la plancha de corcho es el cuadrado de vértices (-1,1), (1,1), (-1,1) y (-1,-1).

1. Piense cómo puede simular el tirar un dardo, ¿qué es necesario generar? Luego piense cómo puede verificar si el dardo cae en el círculo.

2. Escriba una función llamada `tirar_n_dardos(n)` que simula la tirada de `n` dardos y devuelve el cociente entre los que quedaron en el blanco y la cantidad tirada.

3. Invoque a la función con `n` $= 10^8$ y multiplique la respuesta por 4 y piense "¡qué curioso!"

2. El caminante borracho.

Vamos a trabajar con caminos al azar. Una persona en estado de ebriedad se encuentra en un campo y decide comenzar a caminar. Va paso a paso y antes de dar un paso decide al azar en qué dirección lo hará: norte, sur, este u oeste.
Suponga que el borracho comienza en un punto de coordenadas (0,0) y que cada paso que da tiene longitud uno. Así si el primer paso es en dirección norte termina en la coordenada (0,1).

1. Escriba una función `distancia_al_origen(n)` que calcula la distancia al origen luego de haber caminado `n` pasos.

2. Calcule el promedio de repetir 100 veces el experimento para un número de pasos muy grande (empiece con 10^4 y luego haga sufrir a la máquina).

3. Modifique el código para guardar las coordenadas de la posición final de cada uno de los 100 experimentos y realice un gráfico que las muestre.

3. El vendedor viajero.

Vamos a tratar de resolver un clásico problema de optimización. Un vendedor tiene que llevar su mercandería a n ciudades y luego volver a su ciudad de origen. Tiene que pasar una sola vez por cada ciudad. Se quiere encontrar la ruta más corta para realizarlo.

1. Supongamos que la cantidad de ciudades es 4 (la ciudad de origen está incluida) y que las distancias entre las ciudades son las mostradas en la figura 2 (son valores inventados para que se entienda la forma de trabajar). Calcule a mano la ruta más corta. Las siguientes son las 6 rutas posibles comenzando y terminando en 0. Las damos como listas:

```
[0,1,2,3,0], [0,1,3,2,0], [0,2,1,3,0], [0,2,3,1,0], [0,3,1,2,0], [0,3,2,1,0]
```

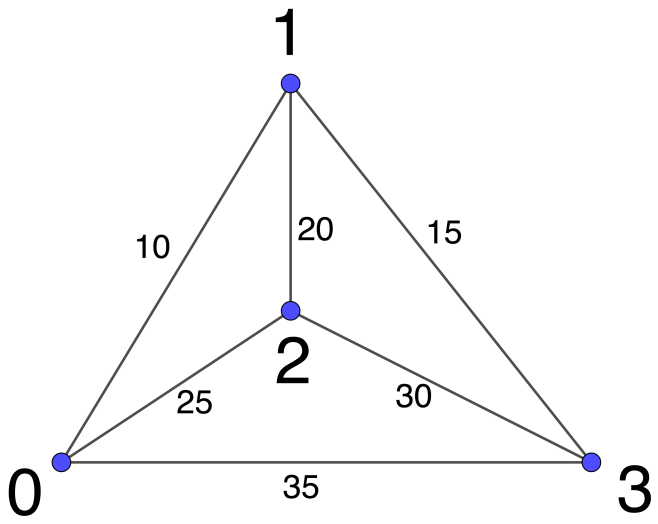

Figura 2: Ciudades y distancias entre ellas

Ahora tratemos de pensar el problema en la computadora. Vamos a suponer que tenemos las coordenadas de las ciudades y que están guardadas en variables. Un recorrido lo pensaremos como una lista de números que comienza y termina en 0 como se mostró antes.

2. Recicle (o vuela a escribir) la función distancia definida en la actividad 4 (la de los planetas) que dadas las coordenadas de dos ciudades devuelva la distancia entre ellas.

3. Defina una función `longitud_de_recorrido(recorrido)` que dado un recorrido devuelve la longitud total del mismo.

4. Defina una función que tome una lista de recorridos (una lista de listas) y devuelva el recorrido con menor distancia. Sugerencia, primero calcule la longitud del primer recorrido. Suponga que es el de menor longitud y luego recorra la lista de los restantes actualizando el de menor longitud.

   Una forma posible

```
def encuentro_el_minimo(lista_de_recorridos):
  recorrido = lista_de_recorridos.pop()
  longitud_minima = longitud_de_recorrido(recorrido)
  recorrido_minimo = recorrido
```

```
    for i in range(len(lista_de_recorridos)):
        ___COMPLETAR___
    return recorrido_minimo
```

5. Ahora trate de hacer un programa que resuelva el problema del vendedor viajero con 5 ciudades. La parte nueva que necesitará es generar la lista de recorridos. Piense cómo hacerlo con la computadora (son 24).

4. La yapa: bugs.

Como en este taller aprendimos a programar no podía faltar el análisis de bugs.
Suponga que hay cuatro bugs en las esquinas de un cuadrado como se ve en a figura. Cada uno de ellos está perdidamente enamorado del que tiene enfrente y deciden los 4 al mismo tiempo comenzar a moverse. Se mueven a la misma velocidad constante (despacito). Si el lado del cuadrado tiene longitud 1, ¿cuánto habrá recorrido cada bug un vez que alcanza a su objeto de deseo?

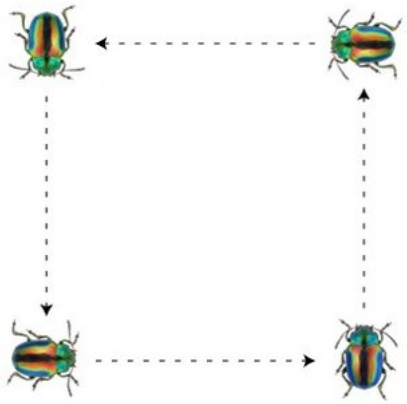

Figura 3: Bugs in love

Este problema ha aparecido varias veces en la sección "A entrenar el cerebro en verano" de diarios de todos el mundo. No nos interesa la respuesta.
Como desafío final les dejamos que hagan un programa que grafique las trayectorias de los 4 bugs. Para ello se sugieren algunas ideas:

- Como vamos a querer graficar, necesitaremos las trayectorias. Guardar las mismas en listas.
- Analizaremos pasos pequeños, que llamaremos `dt` como en la consigna 4. Con `dt=0.01` se obtienen lindos resultados.
- Si `pos_1` y `pos_2` son las posiciones en un determinado momento de los bugs 1 y 2, y suponiendo que 1 persigue a 2 se tiene que la nueva posición del bug 1 es:

  ```
  pos_1 + dt*(pos_2-pos_1)/d(pos_1,pos_2)
  ```

  donde `d(pos_1,pos_2)` es la distancia entre las posiciones.