Universidad Nacional Autónoma de México

Facultad de Ingeniería

Alumno

Barriguete Rodríguez Héctor Alejandro

Sistemas Operativos

Proyecto #1: 21 blackjack

Grupo

Fecha de entrega: 29 de noviembre de 2022

Objetivos

- Desarrollar una problemática que se pueda resolver por medio de programación paralela

Introducción

Se presentará el código del juego 21 blackjack. Este juego presenta muchas reglas y consideraciones, lamentablemente algunas de ellas no se pudieron presentar en la solución propuesta.

Entre las reglas y consideraciones del juego que se pudieron realizar son las siguientes:

1. Se reparten cartas a cada jugador y al Crupier de forma ordenada. Esto quiere decir que se le da una carta a cada jugador y al crupier en cada turno.
2. Se le dan cartas a los jugadores hasta que ellos ya no quieran o superen el puntaje.
3. El Crupier debe de seguir tomando cartas hasta que su puntaje sea igual o mayor a 17.
4. Una vez que todos los jugadores y Crupier se planten (dejen de pedir cartas) se compara el puntaje de cada jugador con el del Crupier.
5. El puntaje del jugador y del Crupier no debe ser mayor a 21, de superarse se pierde automáticamente.
6. El jugador gana si su puntaje es mayor al del Crupier, o si su puntaje es menor al del Crupier siempre y cuando el puntaje del último sea mayor a 21.
7. El jugador pierde si su puntaje es menor al del Crupier, o si su puntaje es mayor a 22 y el del Crupier es menor o igual a 21.
8. El jugador y el Crupier empatan si sus puntajes son iguales o si ambos superan los 21 puntos.
9. Se cuenta con un total de 104 cartas (lo que equivale a dos barajas).
10. Las cartas correspondientes a *K*, *Q* y *J* tienen un valor de 10.
11. El as solo puede tener un valor (a diferencia de la vida real en donde tiene 2 valores) que es 1.

Desarrollo

El lenguaje utilizado para desarrollar el programa es Python es su versión 3.10.0. Las librerías utilizadas son *threading*, *time* y *random*. Se desarrolló y probó en Windows 10.

El programa inicia con la creación de 5 hilos (cuatro de jugadores y uno de Crupier):

```
for i in range(personas_mesa):
    threading.Thread(target=jugador, args=[i]).start()

time.sleep(1)

threading.Thread(target=crupier).start()
```

Para los hilos jugadores, se considera la lista *dinero* (dinero que tiene cada jugador) y *apuesta* (apuesta hecha por cada usuario), para cada jugador se inicializa su respectivo dinero y apuesta:

```
global dinero, apuesta
dinero[id] = 0
apuesta[id] = 0
```

Se entra a un *mutex* en donde se añadirá el id del jugador a la lista *lista_turnos*, se verifica el tamaño de dicha lista, una vez que todos los jugadores hayan agregado su id, se agregará el del Crupier. Posteriormente con la función random se obtiene su dinero y apuesta realizada:

```
mutex.acquire()
lista_turnos.append(id)

if len(lista_turnos) == 4:
    lista_turnos.append(id_crupier)

dinero[id] = random.randrange(4, 12) * 100 #Se genera un número aleatorio
print("Dinero del Jugador #" + str(id) + " $" + str(dinero[id]))

apuesta[id] = random.randrange(1, 4) * 100 #Se genera un número aleatorio
print("Apuesta del Jugador #" + str(id) + " $" + str(apuesta[id]) + "\n")

mutex.release()
```

Finalmente se manda a llamar a la función *mesa*, mandando como argumentos su id, dinero y apuesta.

```
mesa(id, dinero[id], apuesta[id])
```

Para el hilo del Crupier únicamente se manda a llamar a *mesa* y por los argumentos que la función pide se manda su id y ceros para los otros campos:

```
mesa(id_crupier, 0, 0)
```

En *mesa* y omitiendo la inicialización de variables, hay un ciclo *while* que se ejecutará en todo momento, dentro de este lo primero será revisar la lista *lista_plantados*, inicialmente la lista se encuentra de la siguiente manera:

```
lista_plantados = ['S', 'S', 'S', 'S', 'S']
```

La *S* significa que el jugador o Crupier continúa pidiendo cartas, si ya no quiere se cambia por una *P*. La posición se encuentra relacionado con el id del jugador y Crupier, para este último la posición en la lista también es la última.

Se recorre cada elemento de la lista y se hace una cuenta de cuantos están plantados:

```
for plantado in lista_plantados:
    if plantado == 'P':
        contador_plantados += 1
```

Si no todos están plantados se recorre *lista_turnos*, de esta forma se respeta la forma en la que los jugadores fueron llegando, también se comprueba que el jugador de ese id no esté plantado:

```
if contador_plantados != 5: #Si no se plantaron todos
    for turno in lista_turnos: #Verifica el turno del
        if turno == id and lista_plantados[id] == 'S':
```

Después se entra a un *mutex* en donde se obtendrá una carta para darle al jugador, para ello se usa random para el tipo (corazón, diamante, trébol o pica) y su número:

```
tipo = random.randrange(0, 4) # 0
numero = random.randrange(1, 11)
```

Se revisa de quien es el turno (si de un jugador o del Crupier) que su puntaje sea menor a 22 y que quiera otra carta, con un booleano (*carta_repetida*) se determina si dicha carta ya fue jugada dos veces:

```
if (id != id_crupier and int(puntaje[id]) < 22 and pedir_carta) or (id == id_crupier and int(puntaje_crupier) < 22 and pedir_carta_crupier):
    # Si el ID es el de un jugador, su puntaje es menor a 22 y pide carta, o si el ID es el del Crupier, su puntaje es menor a 22 y pide cart

    if(tipo == 0):
        carta_repetida = repartiendo(numero, cartas_corazon, "CORAZONES")
    elif(tipo == 1):
        carta_repetida = repartiendo(numero, cartas_diamante, "DIAMANTES")
    elif(tipo == 2):
        carta_repetida = repartiendo(numero, cartas_trebol, "TREBOLES")
    elif(tipo == 3):
        carta_repetida = repartiendo(numero, cartas_pica, "PICAS")
```

La función *repartiendo* el valor de *carta_repetida*, falso si todavía se encuentra entre las cartas o verdadero si ya no se encuentra en la baraja:

```
def repartiendo(valor, lista_carta, str_carta):
    if (valor in lista_carta): #Si la carta está
        lista_carta.remove(valor) #Se elimina de
        print(valor, "DE", str_carta)
        return False
    else:
        return True
```

Posteriormente, se asigna la carta al jugador o al Crupier, nuevamente verificamos que su puntaje no supere los 21 puntos y quiera otra carta:

```
if id != id_crupier: #Si el ID es el de un jug
    if int(puntaje[id]) < 22 and pedir_carta:
```

Si la carta no ha sido jugada dos veces se le suma al puntaje:

```
if carta_repetida == False:
    puntaje[id] += numero
```

Luego se verifica si dicho puntaje es igual a 21, en caso de serlo el jugador ya no pide una carta:

```
if (int(puntaje[id]) == 21):
    pedir_carta = False
```

En caso contrario se imprime su puntaje y se define si quiere otra carta o no, siempre y cuando su puntaje sea mayor o igual a 11:

```
else:
    print("JUGADOR #" + str(id) + " Puntaje actual:", puntaje[id]

    if int(puntaje[id]) >= 11: #Puntaje mayor/igual a 11
        pedir_carta = probabilidad(int(puntaje[id])) #¿Se pide ot
```

Para determina si quiere o no otra carta se usa la función *probabilidad* en la cual primero se hace una resta entre 21 y el puntaje actual del jugador, luego se obtiene un numero aleatorio entre 1 y 10, si el numero aleatorio es menor o igual al valor de la resta que se hizo anteriormente el jugador pedirá otra carta, en caso contrario se planta:

```
if decision <= diferencia: #Si
    print("Pide otra carta\n")
    return True #Se pide otra c
else: #Si el número aleatorio e
    print("Se planta\n")
    return False #Se planta
```

De esta forma entre mayor sea su puntaje, menor es la probabilidad de que quiera otra carta.

Una vez que ya no quiera otra carta o su puntaje sea mayor a 21, se cambia su estado en *lista_plantados*:

```
if pedir_carta == False or int(puntaje[id]) > 21:
    lista_plantados[id] = 'P' #Se agrega a la list
```

Lo anterior se repetirá hasta que su puntaje sea menor a 22 y quiera otra carta, una vez que alguna de las dos condiciones anteriores no se cumpla se imprime su puntaje final:

```
if int(puntaje[id]) < 22 and pedir_carta: #Si el puntaje es menor a 2

else:
    print("-> JUGADOR #" + str(id) + " Puntaje final:", puntaje[id],
```

Lo anterior solo aplica para los jugadores, en el caso del Crupier se repite casi lo mismo, la única diferencia es que este no se puede plantar hasta que su puntaje sea mayor o igual a 17:

```
if puntaje_crupier >= 17: #Si el
    pedir_carta_crupier = False
```

Al igual que con el jugador, una vez que no quiera otra carta o supere los 21 puntos se cambia su estado en *lista_plantados*:

```
if pedir_carta_crupier == False or int(puntaje_crupier) > 21:
    lista_plantados[id_crupier] = 'P' #Se agrega a la lista de
```

El *mutex* termina aquí:

```
mutex.release()
```

Lo anterior solo aplica cuando es el turno del jugador y no esté plantado, de no ser su turno va a esperar cuatro unidades de tiempo:

```
for turno in lista_turnos: #Verifica el turno del
    if turno == id and lista_plantados[id] == 'S':
    else:
        time.sleep(4) #Espera 4 segundos para volv
```

Una vez que todos los hilos estén plantados se imprime si el hilo jugador le ganó, perdió o empató con el hilo del Crupier:

```
#Se hacen comprobaciones para saber si el jugador gana, pierde o empata con el Crupier
if puntaje[id] < 22 and (puntaje[id] > puntaje_crupier or (puntaje_crupier > 21 and puntaje[id] < puntaje_crupier)):···
elif (puntaje[id] < puntaje_crupier or puntaje[id] > 21) and puntaje_crupier < 22:···
else:
    print("-> Entre el Jugador #" + str(id) + " (" + str(puntaje[id]) +") y el Crupier (" + str(puntaje_crupier) +"), nadie gana\n")
```

Una vez impreso los resultados, se termina el ciclo *while* con un break:

```
break
```

Cabe mencionar que independientemente de cual sea el caso que ocurra, se reinicia *contador_plantados*:

```
contador_plantados = 0
```

Como tal no hay mucha interacción entre los hilos jugadores, la mayor interacción que tienen son las cartas de la baraja que tienen, pues estas son generales para todos. Los hilos jugadores interactúan más con el del Crupier, pues al final se determinará si le ganaron, perdieron o empataron.

A continuación, se muestra la ejecución del programa:

```
Dinero del Jugador #0 $700
Apuesta del Jugador #0 $200

Dinero del Jugador #1 $900
Apuesta del Jugador #1 $300

Dinero del Jugador #2 $500
Apuesta del Jugador #2 $300

Dinero del Jugador #3 $500
Apuesta del Jugador #3 $300

7 DE PICAS
JUGADOR #0 Puntaje actual: 7

5 DE TREBOLES
JUGADOR #1 Puntaje actual: 5

10 DE DIAMANTES
JUGADOR #2 Puntaje actual: 10

3 DE PICAS
JUGADOR #3 Puntaje actual: 3

4 DE DIAMANTES
Crupier Puntaje actual: 4

7 DE PICAS
JUGADOR #0 Puntaje actual: 14

Pide otra carta

2 DE TREBOLES
JUGADOR #1 Puntaje actual: 7

6 DE CORAZONES
JUGADOR #2 Puntaje actual: 16

Se planta

2 DE DIAMANTES
JUGADOR #3 Puntaje actual: 5

3 DE PICAS
Crupier Puntaje actual: 7

10 DE CORAZONES
JUGADOR #0 Puntaje actual: 24

Se planta
```

```
4 DE TREBOLES
JUGADOR #1 Puntaje actual: 11

Pide otra carta

3 DE TREBOLES
JUGADOR #3 Puntaje actual: 8

8 DE TREBOLES
Crupier Puntaje actual: 15

7 DE TREBOLES
JUGADOR #1 Puntaje actual: 18

Pide otra carta

10 DE TREBOLES
JUGADOR #3 Puntaje actual: 18

Pide otra carta

6 DE DIAMANTES
Crupier Puntaje actual: 21

4 DE PICAS
JUGADOR #1 Puntaje actual: 22

Se planta

8 DE PICAS
JUGADOR #3 Puntaje actual: 26

Se planta

-> El Crupier (21) le ganó al Jugador #1 (22)
Dinero actual del jugador: $600

-> El Crupier (21) le ganó al Jugador #3 (26)
Dinero actual del jugador: $200

-> El Crupier (21) le ganó al Jugador #0 (24)
Dinero actual del jugador: $500

-> El Crupier (21) le ganó al Jugador #2 (16)
Dinero actual del jugador: $200
```

Conclusiones

Se logro desarrollar el código de manera satisfactoria, aunque faltan muchas reglas que el juego original contempla, como que el as tiene dos valores, que se puedan jugar más de una ronda. Cabe mencionar que se logro implementar una versión bastante completa del juego.

Considero que la implementación de sincronización es adecuada ya que permite que el juego se desarrolle de manera adecuada, permitiendo que existan turnos y se respete dicho orden en todo momento.

Cabe mencionar que existen pequeños detalles que no se han podido corregir, pero igualmente