Escuela de computación

Ingeniería en computación

Proyecto #3

Estudiante: Jimmy Mok Zheng
Carné: 2018160229

Profesor: William Mata

Fecha de entrega: 27 de mayo del 2018

Contenido

Enunciado del proyecto

En este proyecto, se tendrá que realizar modificaciones y mejoras en el proyecto # 2 (kakuro), por ejemplo:

Botón “Deshacer jugada”: sirve para deshacer jugadas, es decir, revierte las jugadas que se hayan realizado en el tablero.

Botón “Rehacer jugada”: devuelve desde la ultima jugada que haya hecho hasta la primera jugada.

Funcionalidad multinivel: que permite al usuario ir avanzando niveles, solo si completa el nivel.

Funcionalidad para posibles jugadas: muestra todas las combinaciones que se pueden realizar para formar el número clave.

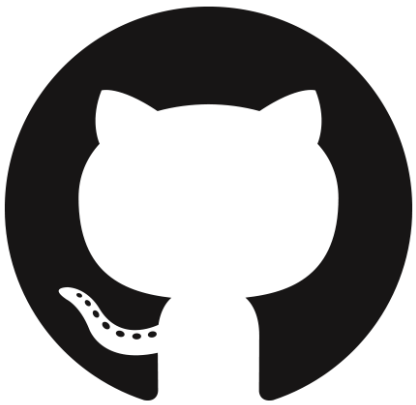
Impresión Top 10: permite obtener la lista del top 10 en PDF.

Temas investigados

Software de control de versiones:

En este proyecto, se utlizo Github como software de control de versiones. Github es una herramienta imprescindible, ya que con la misma se puede hacer una copia del código con el que se esta trabajando, además, se puede tener diferentes versiones del programa para poder tener un respaldo del programa antes de que se realizara alguna modificación, en otras palabras, tiene un sistema seguimiento de incidencias.

Asimismo, este software de control de versiones facilita los trabajos en grupos, ya que los integrantes del grupo pueden realizar cambios al programa que están haciendo. Para poder usar Github, es necesario crear un repositorio (lugar donde se guardará el programa) en el cual se va a subir el programa (rama principal o master) y cuando se realicen cambios en el código, guardar los cambios realizados para tener un registro de las modificaciones, y si se añade nuevas funcionalidades, se crean nuevas ramas, que se van a añadir al master o rama principal, que permiten trabajar en diferentes versiones.



Estructura de datos:

```
def borrar_jugada():
    global jugada
    global lista_jugada
    global contenido
    if len(jugada)==0:
        messagebox.showinfo("Aviso","No has ingresado ninguna jugada para borrar!!! >:v ")# se envia un aviso
    ultima=jugada[len(jugada)-1]
    lista_jugada.append(ultima)
    contenido.append(ultima.get())
    ultima.delete(0,END)
    jugada.remove(ultima)

def rehacer_jugada():
    global lista_jugada
    global contenido
    global jugada
    if len(lista_jugada)==0:
        messagebox.showinfo("Aviso","No hay ninguna por rehacer, ya que no se realizado ninguna!!!")# se envia un aviso
    lista_jugada[len(lista_jugada)-1].insert(0,contenido[len(contenido)-1])#se inserta la ultima jugada en su entry respectivo
    jugada.append(lista_jugada[len(lista_jugada)-1])
    contenido.pop()
    lista_jugada.remove(lista_jugada[len(lista_jugada)-1])
```

En este programa, se utilizaron 3 listas para la función ”rehacer jugada” y “borrar jugada”, en el cual dos de ellas iban intercambiando elemento, ya que si se borraba una jugada, se podía rehacer y viceversa, es decir, se creo una pila de jugadas para rehacer y borrar.

```
 if nivel.get()=="Multinivel":
     if cont_nivel==2:
         partida=eval(lineas[2])
         partida=partida[0]
         cont_nivel=cont_nivel+1
         lista=partida
         celdas(lista)
     if cont_nivel<2:#si es menor a 2, va a entrar al nivel 1 o 2, dependiendo del estado del con
         partida=eval(lineas[cont_nivel])#se saca la partida correspondiente, dependiendo del est
         partida=partida[0]#se saca la primera tabla del nivel correspondiente
         cont_nivel=cont_nivel+1#se suma uno para cuando complete la tabla, se pueda pasar de niv
         lista=partida#se defina la lista con la que se va revisar la tabla
         celdas(lista)#se crea la tabla
     if cont_nivel>2:#si el contador es mayor que 2, el juego se queda en el nivel de 3 neuroas
         partida=lineas[2]#se saca la partida del archivo
         partida=eval(partida)#se quitan los strings
         lista=random.choice(partida)#se saca una partida aleatoria
         celdas(lista)#se crea la tabla
```

Asimismo, al hacer la funcionalidad “Multinivel” se creó un contador para ir pasando la lista que se encontraba en el archivo txt kakuro2018partidas, debido a que cuando se terminaba la tabla, se llamaba a la función que lei los elementos del archivo para sacar una nueva tabla y así sucesivamente.

```
def solo_numeros(casilla):
    punto="."
    barra="/"

    if punto in casilla or barra in casilla:

        result = solo_numeros_aux(casilla,"",0,[],casilla)
        print(result)
        return result

    if punto not in casilla and barra not in casilla:
        return casilla


def solo_numeros_aux(casilla,result,cont,contenedor,casilla_copia):
    if len(casilla_copia)==cont and casilla=="":
        if result!="":
            return contenedor + [result]
    if len(casilla_copia)==cont:
        return contenedor
    if casilla[0].isdigit()==True:
        return solo_numeros_aux(casilla[1:],result+casilla[0],cont+1,contenedor,casilla_copia)
    if casilla[0].isdigit()==False:
        if result!="":
            return solo_numeros_aux(casilla[1:],"",cont+1,contenedor+[result],casilla_copia)
        else:
            return solo_numeros_aux(casilla[1:],"",cont+1,contenedor,casilla_copia)
```

Además, se creo una función recursiva de cola para quitar los signos que indicaban si el número debía ser un botón o un entry, al crear la tabla, ya que estas obstruían en el análisis de los números clave, y se creaba una lista de los números.

```
            return solo_numeros_aux(casilla[1:],"",cont+1,contenedor,casilla_copia)
def permutaciones(m,n):
    if m==2:
        return pares(n)
    else:
        return permutaciones_aux(m,n,[],0,pares(n))

def permutaciones_aux(m,n,lista,cont,ps):

    if cont>len(pares(n)):
        return []
    else:
         x = permutaciones_aux(m-1,ps[cont][0],cont+1,ps[1:])
         return [ps[cont][0]+x[cont][0]] + permutaciones_aux(m-1,ps[cont][0],cont+1,ps[1:])
```

También, se creo una funcion con recursión de pila para sacar las combinaciones de un número, en el cual, se usaba proporciones matemáticas relacionadas con (n , n-1), entre otros.

Funcionalidad extra:

```
def frases():
    global lista_frases
    global cont_frases
    datos= Tk()#se crea una ventana
    datos.title("Datos curiosos")
    datos.geometry("950x100")#Se define el tamaño del menu
    datos.resizable(0,0)#se redimensiona la ventana
    datos.configure(background="gray")#se pone un color a la ventana
    if cont_frases>len(lista_frases)-1:
        cont_frases=0

    curioso=lista_frases[cont_frases]
    cont_frases=cont_frases+1
    texto_frase=Label(datos, text=curioso, fg="black", bg="gray", font="Arial 12").place(x=100,y=30)
```

Se creo una función que proveía datos curiosos o frases en una ventana, mediante un contador que ayudaba a acceder a los elemento de la lista que contenía esta frases.

Conclusiones:

Problemas presentados y soluciones:

Problema 1: no hubo suficiente tiempo, debido a la cantidad de tareas, presentaciones, entre otros, que obstaculizaban el desarrollo del proyecto.

Solución: no hay solución específica, solo dedicar tiempo de la hora de dormir.

Problema 2: se limito muchas de las habilidades del programador, debido a que cada funcionalidad debía ser hecho recursivamente.

Solución: Investigar y pensar nuevas formas.

Problema 3: crear la funcionalidad de combinaciones.

Solución: dedicar más tiempo.

Aprendizajes obtenidos:

En este proyecto se obtuvo una gran cantidad de conocimientos, por ejemplo, que para hacer “rehacer jugada ” y ”borrar jugada” se tiene que hacer que ambas listas se compartan información mientras se ejecutan, es decir, que si se borra una jugada esta ira a la lista de rehacer jugada como una posible jugada a rehacer, ya que fue realizado por el usuario, y viceversa.

Ademas, se comprendio que trabajos que requieren de pocas modificaciones pueden cambiar gran cantidad de la lógica implementada en el programa. Aparte de eso, se adquirieron conocimientos, sobre lo arduo que es hacer mantenimiento de software y su importancia, debido a que el mantenimiento permite que se reduzcan los errores encontrados en los programas y permite al programa actualizarse y adaptarse a la tecnología en la actualidad.

Actividad Realizada	**Horas**
Análisis de requerimientos	10
Diseño de algoritmos	8
Investigación de software de control de versiones	44
Programación	10
Documentación interna	8
Pruebas	6
Elaboración del manual de usuario	2
Elaboración de documentación del proyecto	4.5
TOTAL	92,5

Concepto	**Puntos**	**Puntos obtenidos**	**Avance T/P/N**	**Análisis de resultados**
Posibles jugadas para una clave-cálculos	12		100%	COMPLETADO
Posibles jugadas para una clave despliegue en la GUI	12		100%	COMPLETADO
Deshacer jugada	7		100%	COMPLETADO
Rehacer jugada	7		100%	COMPLETADO
Multinivel (10 puntos por nivel)	30		100%	COMPLETADO
Impresión Top 10	10		10%	Se logro descubrí la librería para el top 10, pero no se pudo implementar a tiempo
Uso de software de control de versiones	15		100%	COMPLETADO
Opción configurar	2		100%	COMPLETADO
Ayuda	5		100%	COMPLETADO
TOTAL	**100**		90%	

Ventana que tira datos curiosos			100%	COMPLETADO