	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	M.C. JOSÉ MAURICIO MATAMOROS DE MARIA Y CAMPOS
Asignatura:	ESTRUCTURA DE DATOS Y ALGORITMOS I
Grupo:	12
No de Práctica(s):	12
Integrante(s):	ARELLANES CONDE ESTEBAN
No. de Equipo de cómputo empleado:	10
No. de Lista o Brigada:	01
Semestre:	2022-2
Fecha de entrega:	23 mayo, 23:59
Observaciones:	

CALIFICACIÓN: _____

Práctica 12: Recursividad

Estructuras de Datos y Algoritmos I

Autor: Arellanes Conde Esteban

1. Objetivo

El alumno aprenderá a modelar problemas de forma recursiva a fin de utilizar el concepto de recursividad en la solución de problemas.

2. Introducción

La recursividad es uno de varios paradigmas de programación que consiste en modelar problemas en base a la repetición y apoyado de otros paradigmas como "*Divide y vencerás*" podemos descomponerlo en problemas más pequeños. Es como poner dos espejos mirándose uno a otro y formándo así un reflejo *ad infitum*.

Una búsqueda exhaustiva consiste en localizar un elemento denominado ("*llave*") dentro de un conjunto. Cuando el conjunto está representado como un arreglo la única forma de localizar una llave en el arreglo es recorrer todo el arreglo de principio a fin comparando uno a uno todos los elementos del conjunto.

El famoso rompecabezas de las torres de Hanoi consiste en solo mover un disco a la vez. Cada movimiento consiste en tomar el disco superior de una de las pilas y colocándolo encima de otra pila, es decir, un disco sólo se puede mover si está encima de una pila. No se puede colocar ningún disco encima de un disco más pequeño. El rompecabezas comienza con los discos ordenados en orden ascendente en uno de los postes formando una pirámide cónica y cuidando que el disco más pequeño esté en la parte superior.

3. Actividad 1.

Con base en el Algoritmo 1 ("*binsearch.py*"), completamos el programa esqueleto e implementamos la función de búsqueda binaria binsearch de forma recursiva. Pruebe su programa con los siguientes conjuntos de datos:

```
python3 binsearch.py 100
python3 binsearch.py 69
```

Complete la siguiente tabla [2 puntos]:

key	**Encontrado**	**Posición**	*h*
29	Sí	1	12
55	Sí	3	11
69	No	No hay	14
100	Sí	5	12
3239	Sí	311	5
12962	Sí	1249	3
44151	Sí	4374	4
50608	Sí	5000	13
74989	Sí	7500	13
99994	Sí	9999	14

4. Actividad 2.

El programa del Apéndice A.2 ("*hanoi.py*") imprime la secuencia de pasos para resolver el rompecabezas de las torres de Hanoi con 3 discos que se quieren mover del poste izquierdo (1) al central (2). Ejecutándolo con la línea: `python3 hanoi.py`

Analizando el programa ("*hanoi.py*") lo modificamos para que imprimiera la secuencia de pasos para resolver el rompecabezas con 5 discos que deben moverse del poste derecho (3) al central (2).

¿Qué modificaciones se realizaron al programa? Explique: [1 punto] Agregamos un contador global para contar el número de pasos dados en cada ejecución y pudimos verificarlo de acuerdo a su desarrollo matemático de $2^n - 1$ donde n es el número de discos y la operación el número de movimientos que tardaríamos en completar el rompecabezas.

¿Cuántos pasos se requieren para mover los 5 discos? Explique: [1 punto] 31 pasos son los que tardaríamos en completar el movimiento de los discos de acuerdo a su desarrollo matemático: $2^{(n)} - 1 => 2^{(5)} - 1 = 32 - 1 = 31$.

¿Cuántos pasos se requerirán para mover 8 discos? Modifique el programa en caso necesario y explique: [1 punto] 255 pasos son los que tardaríamos en completar el movimiento de los discos

de acuerdo a su desarrollo matemático: $2^{(n)} - 1 => 2^(8) - 1 = 256 - 1 = 255$.

5. Actividad 3

El programa del Apéndice A.3 ("*palindrome.py*") determina si las palabras pasadas como argumentos son palíndromos o no, es decir, si las palabras pueden leerse igual de derecha a izquierda que de izquierda a derecha. Ejecutándolo con las líneas:

```
python3 palindrome.py noon
python3 palindrome.py auch
```

Modificando el programa del Apéndice A.3 ("*palindrome.py*") y probando el programa con las siquientes palabras de forma que la función palindrome sea recursiva.

- anilina
- arenera
- erre
- ese
- oso
- rodador
- solos
- someteremos

¿Qué modificaciones realizó al programa para hacerlo recursivo? Explique [5 puntos]: Lo único que necesitamos para hacer el programa recursivo fue adaptar el programa para que llamara a la función dentro de sí misma y con eso fue más que suficiente.

6. Conclusión y Referencias

6.1. Conclusión:

En esta y última práctica comprendimos lo que son y para qué sirven los algoritmos de búsqueda exhaustiva aunado con los paradigmas de recursividad aplicandolos a problemas cotidianos como búsqueda de un valor, palíndromos o el famoso rompecabezas de las torres de Hanoi y des.

6.2. Referencias:

- Thomas H. Cormen. (2009). Introduction to Algorithms 3e. United States of America: Massachusets Institute of Technology.
- Stephen R. Davis. (2015). Beginning Programming with C++ For Dummies. New Jersey, United States of America: John Wiley & Sons.
- Códigos completos: https://www.online-python.com/9JO4smDZxR