Los castillos de Max

Eddy Ramírez Jiménez
Valor 20 %

1. Objetivo

Esta tarea tiene como objetivo que el estudiante demuestre capacidad para la aplicación del manejo de recursión, ciclos, condicionales, operaciones aritméticas y listas en un ambiente de programación.

2. Antecedentes

Existen en programación dos conceptos muy importantes que nos ayudan significativamente a mejorar tiempos de ejecución (al disminuir el número de pasos que requiere un problema) de ciertos programas.

Como primera de estas herramientas son la denominada suma de Gauss que indica lo siguiente:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

La segunda herramienta es la búsqueda binaria, que me permite encontrar sobre un segmento ordenado (o que cumple algún orden total), un valor con un costo logarítmico.

La intención de este proyecto es poder utilizar esos dos conceptos juntos para poder resolver un problema de forma eficiente.

3. Especificación del problema

Resulta que un "castillo" de naipe se puede hacer de varios niveles. Un ejemplo gráfico se puede ver en la figura 1 de la página 1. En donde se pueden apreciar cómo se deben colocar los castillos para un castillo de 1 nivel, 2 niveles y 3 niveles respectivamente.

3.1. Armando castillos

A Max, un estudiante de la carrera de ingeniería en computación de la sede de Alajuela, le han dado cierta cantidad c de naipes, y él quiere hacer el castillo más grande posible (de más niveles) con los naipes que le han sido otorgados. Luego, con los que le queden (si quedara alguno) quiere hacer lo mismo y así sucesivamente hasta que ya no le queden naipes suficientes para hacer más castillos.

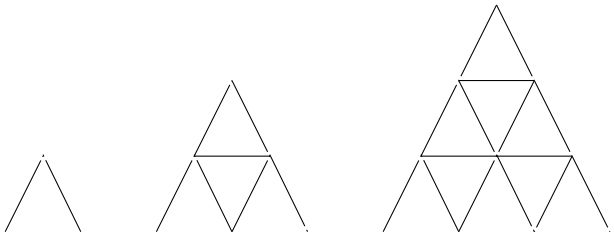

Figura 1: Castillos de 1,2 y 3 niveles respectivamente

Dado que él está muy ocupado haciendo un intérprete de *Brainfuck* o buscando donas con pasas, le ha pedido a cada uno de ustedes que haga un programa que le indique cuáles son los castillos que puede armar con la cantidad de naipes que le son dados. Todos los castillos de n niveles, deben tener la misma cantidad de naipes (es decir, no se puede construir un castillo si le falta una sola carta).

Además, su programa debe ser muy eficiente en indicar cuáles son los castillos que pueden hacer dada la cantidad de cartas iniciales de Max.

4. Entradas y salidas

Las entradas serán por consola (utilizando la función *input*() de Python3) y **no** debe realizar ningún tipo de validación.

4.1. Entrada

La entrada consistirá en un entero T, que va indicar la cantidad de casos de prueba a los que será sometido el programa. Puede tener la certeza que $1 \leq T \leq 10^5$.

Las siguientes T líneas tienen cada una un entero C ($1 \leq C \leq 10^{10}$) indicando la cantidad de naipes que tiene Max para hacer sus castillos de naipe.

4.2. Salida

Por cada caso de prueba se imprime una línea. En cada línea deben ser impresas, separadas por espacio, las alturas (de mayor a menor) de los castillos de naipes que Max puede hacer. Después del último castillo, no se debe dejar espacio.

4.3. Ejemplos de entrada y salida

Entrada 1

```
5
2
1
9
30
31
```

Salida 1

```
1

2 1
3 3
3 3
```

5. Metodología

Debe de escribir las funciones con cuidado y documentar los pasos que haya ejecutado para poder lograr la completitud del programa.

6. Rúbrica

La evaluación va a estar constituida por dos partes, por un lado la parte p que es el programa y la parte d que es la documentación. La nota de su proyecto será $\sqrt{d \times p}$ (a esto se le denomina promedio geométrico).

6.1. El programa

A continuación la evaluación del programa:

Producto esperado	. % .
Uso correcto de la suma de Gauss	10 %
Uso correcto de la búsqueda binaria	20 %
Correctitud de la salida	70 %
TOTAL	100 %

Su programa será sometido a varios archivos de entrada (con diferentes casos de prueba), esto va a determinar la correctitud de la salida. En cada archivo se tomará el tiempo que tarda en responder ese bloque de casos de prueba. Se otorgará un cero en donde:

- La entrada no se lee de forma correcta
- El programa tarda más de un minuto en resolver el caso de prueba
- Se imprimen más textos que los especificados en este documento

La nota de un bloque de casos de prueba, será la cantidad de líneas correctamente mostradas entre la cantidad de líneas correctas totales (en la posición esperada).

6.2. Documentación

Toda la documentación debe estar hecha en LaTeX o de lo contrario, no será recibida (o se otorgará nota de cero). Además, debe tener un margen máximo de 2.5 cm por cada lado y una letra de tamaño 11 o 12 pts.

Tener presente que la ortografía resta 1 punto por cada falta y no tener bibliografía implica una nota de cero.

Producto	**Valor**
Portada	5 %
Resumen ejecutivo	10 %
Introducción	5 %
Marco teórico	10 %
Descripción de la solución	30 %
Resultados de pruebas	20 %
Conclusiones	10 %
Aprendizajes	10 %
Total	100 %

A continuación se detalla cada parte de la documentación:

- Portada

  Con el logo del TEC, título del proyecto, nombre del curso, nombre de los estudiantes, nombre del profesor y fecha.

- Resumen ejecutivo

  En una página **(no más, no menos)** un resumen del proyecto, su solución y resumen de los resultados de las pruebas.

- Introducción

  Una descripción del documento y en qué consistía el proyecto.

- Marco teórico

  Una descripción del lenguaje utilizado (antecedentes, historia), una breve documentación de las bibliotecas utilizadas y de los IDE's utilizados. También información sobre la suma de Gauss y búsqueda binaria.

- Descripción de la solución

  Detalle del algoritmo de solución para cada problema y subproblemas encontrados durante el desarrollo del proyecto y su respectiva implementación.

- Resultados de pruebas

  Una vez finalizada la codificación del proyecto, debe de realizar diversos casos de prueba, tomar tiempos de ejecución (usando el comando time) y generar gráficas de rendimiento. Considerar probar los márgenes con exhaustividad, de acuerdo con los límites de la especificación.

  Para justificar los tiempos sería bueno especificar el tipo de hardware en que se corrió y en la medida de lo posible no correr nada más en la computadora en ese momento.

- Conclusiones

  Un análisis de los resultados y del proyecto en general. Así como una explicación teórica de los resultados. ¿Considera que se puede ejecutar más rápido de los valores obtenidos?

- Aprendizajes

  De manera **individual**, qué aprendió cada integrante de los equipos con la realización de este proyecto.

- Bibliografía (en formato IEEE, preferiblemente)

7. Estimación de tiempo

- Se espera que el programa se realice rápidamente y que se trabaje por a lo sumo 6 horas.

8. Aspectos Administrativos

- El trabajo debe de ser realizado de forma **individual**. Este será el último proyecto que se deberá trabajar de esta forma, a partir del segundo proyecto, se podrá hacer en tríos.
- Toda la programación debe realizarse en Python3 y será revisado sobre Linux.
- La fecha límite de entrega de este proyecto es el 31 de marzo de 2023 a las 11:59 pm.
- El medio de entrega será por correo electrónico al correo de Eddy en edramirez@itcr.ac.cr[1].
- El nombre del archivo deben ser las seis primeras letras del apellido seguido del primer nombre. Si el apellido fuera menor, se usa todo el apellido. Ejemplo: *ramireeddy.py* y debe entregarse en un .tgz o .rar .
- El nombre del archivo .pdf que contenga la documentación debe ser el mismo que del código, pero .pdf, ejemplo *ramireeddy.pdf*

[1] El asunto debe ser: **IC-1803 - Taller - Proyecto1 [Nombre del estudiante]**