Universidad Simón Bolívar
División de Ciencias Físicas y Matemáticas
Departamento de Computación y Tecnología de la Información
CI5651: Diseño de Algoritmos I

Profesor: Ricardo Monascal. **Estudiante**: Astrid Alvarado, 18-10938.

Tarea 1 (9 %):

1. Pregunta 1

Suponga que `bogoMin` recibe un arreglo a de tamaño n.

- **Cota Inferior (Mejor Caso)**
  Para este análisis, se va a suponer que el primer elemento (llámese `x`) del iterador `permutaciones(a)` es el arreglo ya ordenado de menor a mayor.

  En este caso, se tendrá entonces que el predicado `ordenado(x)` evaluará `true` en la primera iteración del ciclo. Dado que el mismo realiza esta verificación en tiempo lineal, se tiene entonces que `bogoMin` $\in \Omega(n)$
- **Cota Superior (Peor Caso)**
  Para este análisis, se va a suponer que el último elemento (llámese `x`) del iterador `permutaciones(a)` es el arreglo ya ordenado de menor a mayor, en donde todos sus elementos además no están repetidos, por lo que la cantidad total de permutaciones resulta en $n!$

  En este caso, se tendrá entonces que el predicado `ordenado(x)` evaluará `true` en la última iteración del ciclo. Dado que el mismo realiza esta verificación en tiempo lineal, se tiene entonces que `bogoMin` $\in O(n! \cdot n)$

  Note que, en caso de que en el arreglo existieran elementos repetidos, el número de permutaciones del mismo resultaría en:

  $$\frac{n!}{m_1!m_2!\cdots m_k!}$$

  Donde $m_1,\ m_2, \ldots,\ m_k$ corresponde al número de repeticiones de cada elemento donde además $m_1 + m_2 + \cdots + m_k = n$. Sin embargo, este no se considera el "peor caso" dado que para todo n se cumple:

  $$\frac{n!}{m_1!m_2!\cdots m_k!} \leq n!$$

  Siendo así que el peor caso es considerar todos los elementos distintos entre sí.

2. Pregunta 2

Se define Φ como el **número de personas con el sombrero puesto**, el cual representa el peor caso para la función `sombrerear()`.

Se tiene entonces que, para la fila de N personas:

- **Caso A:** `sombrerear()` se detiene en la k-ésima persona, con $k < N$. Esto significa que $k-1$ personas tenían el sombrero puesto y se lo quitaron, y la k-ésima persona lo tenía quitado y se lo puso.

  En este caso, se tendrá que el costo real (c_i) será la cantidad total de sombreros, tanto quitados como puestos, por lo que $c_i = k$. Además, se tiene que $\Phi(E_i)$ representará la k-ésima persona, por lo que $\Phi(E_i) = 1$ pues la k-ésima persona tiene el sombrero quitado y luego de `sombrerear()` lo tendrá puesto. Por otro lado, $\Phi(E_{i-1})$ será las $k-1$ personas en la fila que tenían el sombrero puesto, por lo que antes de `sombrerear()` resulta en $\Phi(E_{i-1}) = k-1$.

  Teniendo esto en cuenta, el costo amortizado será:

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(E_i) - \Phi(E_{i-1}) \\ &= k + 1 - (k-1) \\ &= \cancel{k} - \cancel{k} + 2 = 2\end{aligned}$$

- **Caso B:** `sombrerear()` se detiene en la N-ésima persona. Esto ocurre solo si todas las N personas tenían el sombrero puesto al inicio, por lo que todas se lo quitan y el proceso se detiene al terminar la fila. Para este caso, se tendrá entonces que $c_i = N$, $\Phi(E_i) = 0$ dado que luego de `sombrerear()` ninguna persona termina con el sombrero puesto, y finalmente $\Phi(E_{i-1}) = N$ puesto que es la cantidad de personas que tenían el sombrero puesto antes de `sombrerear()`.

  Con esto, el costo amortizado será:

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(E_i) - \Phi(E_{i-1}) \\ &= \cancel{N} + 0 - \cancel{N} = 0\end{aligned}$$

En ambos casos, se puede ver que el costo amortizado resultante es 2 y 0 respectivamente, y como es constante, se concluye entonces que en el peor de los casos el costo amortizado es $O(1)$ de cada llamada individual. ■

3. Pregunta 3

Se quiere demostrar que $\oplus-$SAT pertenece a P. Para esto se debe encontrar un algoritmo determinista que resuelva el problema en tiempo polinomial.

Se sabe que $\oplus-$SAT es una modificación de SAT, por lo que se quiere buscar una asignación de variables booleanas tal que toda la fórmula evalúe verdadero. Además, cada fórmula en $\oplus-$SAT sigue la siguiente estructura:

$$C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

Donde cada cláusula es de la forma $C_i = l_{i_1} \oplus l_{i_2} \oplus \cdots \oplus l_{i_k}$. Para que toda la fórmula evalúe verdadero, es necesario que cada cláusula C_i sea verdadera, por lo que, teniendo esto en cuenta, se propone hacer una reducción del problema a un sistema de ecuaciones líneal donde la solución x del mismo sea tal que $x \in \{0,1\}^n$ (donde 0 representa a `false` y 1 `true`)

Para lograr esto, se hará la siguiente traducción de las cláusulas a ecuaciones líneales:

- Las disyunciones exclusivas $\oplus$ se van a tratar como la suma módulo 2 ($+_2$), dado que ambos operadores generan el mismo comportamiento.
- Cada literal de la forma l_i se transforma a una variable x_i
- Cada literal de la forma $\neg l_i$ se transforma en $1 +_2 x_i$, pues ambas expresiones presentan el mismo comportamiento.
- Se podrá simplificar la fórmula siguiendo las propiedades de la aritmética modular.

Con esto, una expresión del tipo $l_1 \oplus \neg l_2 \oplus l_3$ será traducido a la ecuación $x_1 +_2 (1 +_2 x_2) +_2 x_3$. Por otro lado, dado que se quiere que cada cláusura sea verdadera, dicha ecuación deberá ser exactamente igual a 1, por lo que la ecuación resultante será:

$$
\begin{aligned}
& x_1 +_2 (1 +_2 x_2) +_2 x_3 = 1 \\
\equiv\ & x_1 +_2 x_2 +_2 x_3 = 0 \qquad \leftarrow \text{(Simplificación: } +_2 \ 1 \text{ en ambos lados de la igualdad)}
\end{aligned}
$$

Con esto se tiene un algoritmo de traducción que toma $O(k)$, donde k es la longitud de la fórmula. Luego, para encontrar los valores para cada variable tal que dicha expresión evalúe a verdadero, basta entonces con resolver el sistema $Ax = b$ mediante reducción Gaussiana, en donde las operaciones permitidas serán:

- Intercambio de filas.
- Sumar ($+_2$) dos filas.

Así, al llevar la matriz aumentada $[A|b]$ a una matriz escalonada, el algoritmo será capaz de encontrar una inconsistencia (en donde se dirá que la fórmula es insatisfacible) o al encontrar una o varias soluciones al sistema de ecuaciones.

Dado que se aprovecha de la eliminación Gaussiana para resolver este problema, se tiene que el tiempo de complejidad para determinar la satisfacibilidad de la fórmula será $O([\max(m, n)]^3)$, donde m, n son las dimensiones de la matriz. Dado que este algoritmo es determinista y se resuelve en tiempo polinomial (pues tanto la traducción como la resolución del sistema de ecuaciones toma tiempo polinomial), se tiene que $\oplus-\text{SAT} \in P$ ■