UNIVERSIDAD SIMÓN BOLÍVAR
CI-5651 - Diseño de Algoritmos I
Prof. Ricardo Monascal

Resuelto por:
Christopher Gómez

Tarea 1: Complejidad algorítmica y análisis amortizado

1. Considere el algoritmo `stupidSort` para ordenar un arreglo de menor a mayor:

```
def stupidSort(a):
    for x in permutaciones(a):
        if ordenado(x):
            return x
```

Puede suponer que `permutaciones` es un iterador que devuelve, una a una, todas las permutaciones de un arreglo en tiempo constante. Además, `ordenado` es un predicado que verifica si un arreglo está ordenado de menor a mayor en tiempo $\Theta(n)$.

Diga cotas inferior y superior para `stupidSort` (correspondiendo al peor y mejor caso, respectivamente). Esto es, defina dos funciones f y g tal que se cumpla que:

- `stupidSort` $\in O(f)$
- `stupidSort` $\in \Omega(g)$

Para el mejor caso, supongamos que el primer elemento que retorna el iterador `permutaciones` es el arreglo ordenado. En este caso, el algoritmo `stupidSort` retorna el arreglo ordenado en tiempo $\Theta(n)$, que es el tiempo que tarda `ordenado` en verificar que el arreglo está ordenado. Por lo tanto, $g(n) = n$.

Por otro lado, el peor caso se da cuando el arreglo ordenado resulta ser el último retornado por permutaciones, lo cual solo ocurre cuando el iterador ha agotado todas las permutaciones posibles sobre un arreglo de n elementos, que son $n!$. Para cada una de estas permutaciones `ordenado` verifica si está ordenado en $\Theta(n)$, por lo que el tiempo total es $O(n! \times n)$. Así, $f(n) = n! \times n$.

2. En la sede del banco caben hasta C personas. La sede empieza vacía y los clientes entran uno a uno. Cuando la capacidad de la sede es superada, la misma cierra y la totalidad de los clientes es transferida a una sede más grande, con capacidad $2 \times C$.

Nótese que si contamos cuántas personas deben trasladarse al entrar un cliente a la sede, ocurre uno de dos casos:

- Si aún no se ha superado la capacidad de la sede, la cantidad de personas que debe trasladarse es $O(1)$ — únicamente el cliente que entra.
- Si se supera la capacidad de la sede, la cantidad de personas que debe trasladarse es $O(C)$ — todos los clientes en la sede del banco más el cliente nuevo.

Demuestre que el orden amortizado, en el peor caso, para el ingreso de un cliente es $O(1)$.

Para el análisis amortizado se usará el método del potencial. Para ello, definamos el potencial $\Phi(S_i) = 2n_i - C_i$, donde n_i es la cantidad de clientes en la sede del banco y C_i es la capacidad de la sede, para un estado S_i del banco. Así, tenemos dos casos para el ingreso de un cliente:

- Para $n_i \leq C_{i-1}$ no nos interesa el costo amortizado, ya que el costo real es $O(1)$ y no es el peor caso.
- Para $n_i > C_{i-1}$ se da el peor caso, en el que se supera la capacidad de la sede y se deben trasladar los n_i clientes a una sede con capacidad $2 \times C_{i-1}$. Tenemos entonces que $C_i = 2 \times C_{i-1}$ y $C_{i-1} = n_i - 1$. Veamos el costo amortizado:

$$
\begin{aligned}
\hat{c_i} &= c_i + \Phi(S_i) - \Phi(S_{i-1}) \\
&= n_i + (2n_i - C_i) - (2(n_i - 1) - C_{i-1}) && \text{(Def. de } \Phi \text{ y } C_i, C_{i-1}) \\
&= n_i + \cancel{2n_i} - 2C_{i-1} - (\cancel{2n_i} - 2 - C_{i-1}) && \text{(Aritmética, } C_i = 2C_{i-1}) \\
&= n_i - C_{i-1} + 2 && \text{(Aritmética)} \\
&= \cancel{n_i} - (\cancel{n_i} - 1) + 2 && (C_{i-1} = n_i - 1) \\
&= 3 && \text{(Aritmética)} \\
&= O(1)
\end{aligned}
$$

  Luego, veamos que el potencial inicial es $\Phi(S_0) = -C_0$, suponiendo que la sede empieza vacía y la capacidad inicial es C_0; y para cualquier estado posterior S_i se cumple que $\Phi(S_i) \geq -C_0$, ya que en el mejor caso $\Phi(S_i) = \Phi(S_{i-1}) + 2$, y en el peor caso (al superar la capacidad) $\Phi(S_i) = 2n_i - n_i + 1 = n_i + 1 > -C_0$. Por lo tanto, la función de potencial es correcta y hemos demostrado que el costo amortizado para el ingreso de un cliente en el peor caso es de $O(1)$.

3. El problema $\oplus-$SAT modificación sobre el problema de satisfacibilidad booleana, donde la fórmula proposicional está en forma normal conjuntiva, pero cada literal está separado por disyunciones exclusivas (en vez de disyunciones tradicionales).

   Recordemos que $P \oplus Q$ es *cierto* si y solo si uno de entre P y Q es *cierto* (el otro siendo falso).

   Diga si $\oplus-\text{SAT} \in P$ o si $\oplus-\text{SAT}$ es $NP-$completo. Demuestre su afirmación.

   Se tiene que $\oplus-\text{SAT} \in P$. Para demostrarlo, basta con construir un algoritmo que resuelva el problema en tiempo polinomial, veamos algunas ideas para construirlo:

   - Sea C una cláusula, esta tiene la forma $C = l_1 \oplus l_2 \oplus \cdots \oplus l_n$, donde cada l_i, $1 \leq i \leq n$, es un literal.
   - Cualquier cadena dentro de la cláusula C conformada por literales que sean $false$ evalúa a $false$. Es decir, $false \oplus false \oplus \cdots \oplus false \equiv false$.
   - Al introducir un $true$ en una cadena, el resultado de la evaluación cambia a $true$. Es decir, $false \oplus \cdots \oplus false \oplus true \oplus false \oplus \cdots \oplus false \equiv true$.
   - Al introducir otro $true$ en la cadena, el resultado de la evaluación vuelve a cambiar a $false$, ya que $true \oplus true \equiv false$.

- De las tres observaciones anteriores, se puede derivar que la evaluación de una cláusula C solo depende de la cantidad de literales $true$ que tenga. Si la cantidad de literales $true$ es par, la evaluación es $false$, y si es impar, la evaluación es $true$.

Teniendo en cuenta lo anterior, que deriva de la definición y propiedades del operador $\oplus$, podemos plantear el problema como buscar una asignación en la que la cantidad de literales $true$ en cada cláusula sea impar, y para resolver este problema construiremos un sistema de ecuaciones lineales de la siguiente forma:

- Sea P una fórmula proposicional en forma normal conjuntiva, con n cláusulas y m átomos, cada cláusula C_i tiene la forma $C_i = l_{i1} \oplus l_{i2} \oplus \cdots \oplus l_{ij_i}$, donde $1 \leq i \leq n$ y $j_i \leq m$ es la cantidad de literales en la cláusula C_i.
- Sea A una matriz de $n \times m$, donde cada fila i representa una cláusula C_i y cada columna j representa un átomo a_j. El valor de la entrada A_{ij} es 1 si el átomo a_j aparece en la cláusula C_i, y 0 en caso contrario.
- Sea b un vector de n elementos, donde cada elemento i representa una cláusula C_i. El valor del elemento b_i es 1 si la cláusula C_i tiene una cantidad par de literales negados, y 0 en caso contrario.

Con lo anterior, podemos construir un sistema de ecuaciones lineales de la forma $Ax \equiv b \pmod 2$, donde x es un vector de m elementos, y cada elemento x_j representa un átomo a_j. Si el sistema tiene solución, cada elemento x_j de la solución representa la asignación del átomo a_j, siendo $a_j \equiv true$ si $x_j \equiv 1 \pmod 2$, y $a_j \equiv false$ si $x_j \equiv 0 \pmod 2$. En caso contrario, el sistema no tiene solución y la fórmula proposicional P no es satisfacible.

Como se puede ver, el sistema de ecuaciones lineales se puede construir en tiempo polinomial, ya que solo se requiere recorrer la fórmula proposicional P una vez para construir la matriz A y el vector b, lo cual se puede hacer en tiempo $O(nm)$. Por otro lado, el sistema de ecuaciones lineales se puede resolver en tiempo polinomial, ya que se puede aplicar el algoritmo de eliminación Gaussiana en módulo 2 (cada operación se hace tomando el módulo 2 del resultado), que tiene complejidad $O(n^3)$. Por lo tanto, el problema $\oplus-$SAT se puede resolver de forma exacta en tiempo $O(n^3 + nm)$, quedando demostrado que $\oplus-\text{SAT} \in P$.