Ejercicio resuelto en clase en semana 2

Ejercicio 1 (Algoritmo GS con función de comparación de propuestas)**.** Considere el algoritmo de Gale-Shapley para formar un emparejamiento estable entre dos conjuntos, $M = \{m_1, m_2, \dots, m_n\}$ y $W = \{w_1, w_2, \dots, w_n\}$. En la versión presentada en la figura 1.1, las preferencias de los elementos de M están determinadas por listas de preferencias que constituyen parte de la entrada del algoritmo (igual que en el libro de referencia), pero las preferencias de los elementos de W se resuelven en tiempo de ejecución mediante invocaciones a una función de booleana, $pref$. Para $w \in W$, $m \in M$ y $m' \in m$, el resultado de $pref(w, m, m')$ es **true** si w prefiere a m antes que a m' y **false** en caso contrario.

Algorithm *Gale-Shapley*
    Inicialmente p está libre para todo $p \in M \cup W$
    while *existe* $m \in M$ *libre que no se ha propuesto a todo* $w \in W$ **do**
        Sea w el elemento de W de mayor preferencia para m al cual m no se ha propuesto
        if w *está libre* **then**
            Emparejar m con w
        else
            Sea m' la actual pareja de w
            if $pref(w, m, m')$ **then**
                Separar a w de m' y emparejar m con w
            else
                w rechaza a m

Figura 1.1: Algoritmo para formar un emparejamiento estable.

(a) Supongamos que para toda instancia del problema se cumple que $pref(w, m, m')$ requiere tiempo $O(n)$ para $w = w_1$ y requiere tiempo $O(1)$ para $w \in W \setminus \{w_1\}$. Muestre que este algoritmo admite una implementación cuyo tiempo de ejecución es $O(n^2)$.

(b) Generalizando el escenario anterior, supongamos ahora que para toda instancia del problema existe un subconjunto $\bar{W} \subseteq W$ tal que

$pref(w, m, m')$ requiere tiempo $O(n)$ para $w \in \bar{W}$ y requiere tiempo $O(1)$ para $w \in W \setminus \bar{W}$, donde $|\bar{W}| \leq \epsilon|W|$ para cierta constante positiva ϵ. ¿Podemos afirmar que este algoritmo admite una implementación cuyo tiempo de ejecución es $O(n^2)$?

(c) En las mismas condiciones que la parte anterior, ¿podemos afirmar que $n^2 = o(T(n))$, donde $T(n)$ denota el tiempo de ejecución de este algoritmo?

Recordar: $f(n) = o(g(n))$ si y solo si $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$.