1. Entre Mezclas e Inserción yo elegiría Mezclas por las siguientes razones:
    a. El tiempo de Mezclas es $\Theta(n\log n)$ es decir, su mejor, peor y caso promedios son $\Theta(n\log n)$, mientras que Inserción en su peor y caso promedio son $\Theta(n^2)$, que es mucho mayor.
    b. Aunque el mejor caso de Inserción es $\Omega(n)$, y es mucho mejor que $\Omega(n\log n)$ de Mezclas, es muy fácil identificar si una lista está ordenada en tiempo $\Theta(n)$, por lo que podría identificar si el arreglo ya estaba ordenado muy rápidamente, y decidir NO aplicar ningún algoritmo.
    c. Mezclas, aunque mucho más rápido que Inserción, ocupa mucho espacio de memoria y hay algoritmos mejores y que también usan $O(n\log n)$, pero si sólo tenemos estas dos opciones, aún así Mezclas gana.
2. P.D. $5n^2 - 20n + 19 = \Theta(n^2)$
    a. Tenemos que ver que existen $c_1>0$, $c_2>0$ y $n_0>0$ tales que:
    $0 \leq c_1 n^2 \leq 5n^2 - 20n + 19 \leq c_2 n^2$
    b. Dividiendo las desigualdades por n^2, tenemos:
    $c_1 \leq 5 - 20/n + 19/n^2 \leq c_2$
    c. De la primera desigualdad vemos que desde $n_0=1$, el valor de la derecha, empezando en 4, aumenta hasta llegar a 5 cuando n se acerca a infinito. Así el valor que se puede proponer para c_1 es 4 y el valor para c_2 es 5.
3. P.D. $3n^2 \neq o(n^2)$
    a. Por contradicción, supongamos que **para toda** $c>0$, existe $n_0>0$, tales que:
    $0 \leq 3n^2 < c*n^2$
    b. Dividiendo entre n^2 tenemos:
    $3 < c$
    c. Esto nos esta forzando a que para que se cumpla la desigualdad, c tiene que ser mayor que 3, o sea si tomamos $c=1$, la desigualdad no se cumple, para ninguna n_0. Lo cual es una contradicción, por lo que se tiene el resultado.
4. P.D. $2^{n+k} = O(2^n)$
    a. Mostraremos que existen $c>0$ y $n_0>0$, tales que:
    $0 \leq 2^{n+k} \leq c*2^n$ para toda $n > n_0$
    b. Dividiendo entre 2^n, tenemos:
    $2^k \leq c$
    c. Proponemos así $c=2^k$
    d. Como lo anterior se cumple para cualquier n, proponemos el valor $n_0=1$.
5. $O(f(n))$ incluye funciones que están "por abajo" de $f(n)$, pero $o(f(n))$ contiene funciones que están MUY POR ABAJO de $f(n)$. Por lo que un algoritmo que se tarde $o(f(n))$ será mucho más rápido que uno que se tarde $O(f(n))$.
6. Si a y b son dos valores positivos, $a \leq \max\{a, b\}$ y $b \leq \max\{a, b\}$
    a. Además, por la 1ª desigualdad tenemos:
    $a * b \leq b * \max\{a, b\}$
    b. Y por la 2ª desigualdad también tenemos:
    $b * \max\{a, b\} \leq \max\{a, b\}*\max\{a, b\} = \max\{a, b\}^2$

c. Así $a * b \leq \max\{a, b\}^2$
d. Como esta última desigualdad es cierta siempre que a y b sean positivos, sean c=1 y n_0=1
e. Tenemos $f(n) * g(n) \leq \max\{f(n), g(n)\}^2$, para toda n
f. Por lo que $f(n) * g(n) \leq c * \max\{f(n), g(n)\}^2$, para toda $n > n_0$
g. Por lo tanto $f(n) * g(n) = O(\max\{f(n), g(n)\}^2)$