```
# ===================================== #
# TRI FUSION                            #
# Programme de base avec temps de calcul. #
# Version du 25/11/2015                 #
# ===================================== #

# Importations
# ============
from time import perf_counter
from random import randint

# La fonction de fusion
# =====================
# Cette fonction reçoit en argument deux listes TRIEES L1 et L2 et renvoie
# une liste L triée contenant tous les éléments de L1 et L2.
def fusion(L1,L2):
    # Initialisations
    L = []
    # On vide une des deux listes (mais on ne sait pas quelle liste sera vidée
    # la première !)
    while L1 and L2:
        if L1[0] < L2[0]:
            L.append(L1.pop(0))
        else:
            L.append(L2.pop(0))
    # On renvoie la concaténation de L, L1 et L2 sachant que L1=[] ou L2=[]
    return L + L1 + L2

# La fonction de tri
# ==================
def tri_fusion(L):
    l = len(L)
    if l <= 1:
        return L
    else:
        m = l//2
        return fusion(tri_fusion(L[:m]),tri_fusion(L[m:]))

# Génération d'une liste de nombres entiers aléatoires
# ====================================================
n = int(input('Longueur de la liste à trier ? '))
valmax = int(input('\nValeur maxi des nombres générés ? '))
LNA = [randint(0,valmax) for i in range(n)]
L_init = list(LNA)

# Tri de la liste LNA des nombres aléatoires
# ==========================================
t_start = perf_counter()
LT = tri_fusion(LNA)
t_end = perf_counter()
t_ellapse = t_end - t_start

# Affichage des listes (initiale et triée) ...
# ...si elles ne sont pas trop grandes ! :)
# ==========================================
if n <= 50:
    print('\nListe initiale :')
    print(L_init)
    print('\nListe triée :')
    print(LT)

# Affichage de la durée du tri
# ============================
print('\nDurée du tri : ',t_ellapse)

# ===============
# FIN DU PROGRAMME
```