```
# ========================================================== #
# TRI RAPIDE (QUICK SORT)                                    #
# Evaluation expérimentale de la complexité moyenne.         #
# (nombre de comparaisons effectuées).                       #
# Version du 21/11/2015                                      #
# ========================================================== #

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

# La fonction de tri
# ==================
def tri_rapide(L):
    global c
    lL = len(L)
    if lL <= 1:
        return L
    else:
        # on choisit ici comme pivot l'élément "central" de la liste
        x = L.pop(lL//2)
        Lg, Ld = [], []
        # La condition du "while" est satisfaite tant que la liste L n'est pas vide
...
        while L:
            c += 1
            y = L.pop(0)
            if y < x:
                Lg.append(y)
            else:
                Ld.append(y)
        return tri_rapide(Lg) + [x] + tri_rapide(Ld)

# Saisie des paramètres et initialisations.
# La variable globale c pour compter le nombre de comparaisons effectuées
# ========================================================================
c = 0
ntris = int(input('Combien de tris souhaitez-vous ? '))
n = int(input('\nLongueur des listes à trier ? '))
valmax = int(input('\nValeur maxi des nombres générés ? '))
t_ellapse_total = 0

for i in range(ntris):
    LNA = [randint(0,valmax) for i in range(n)]
    t_start = perf_counter()
    LT = tri_rapide(LNA)
    t_end = perf_counter()
    t_ellapse_total += t_end - t_start

# Affichage du temps de tri sur nxln(n) moyen qui doit être à peu près constant
(entre 1,5 et 2)
#
=============================================================================================
=========
print('\nDurée du tri : ',t_ellapse_total/(ntris*n*log(n)))
print('\nNombre total de comparaisons : ',c)
print('\nComparaisons : ',c/(ntris*n*log(n)))

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