```python
# Convention : le plus petit est l'élément de rang 0.

def card(X):
    return len(X)

def nPetits(p, X):
    nb = 0
    for k in range(len(X)):
        if X[k] < p:
            nb += 1
    return nb

def nPetits_1(p, X):
    return len([x for x in X if x < p])

def partitionP(p, X):
    L = []
    for u in X:
        if u < p:
            L.append(u)
    return L

def partitionP_1(p, X):
    return [x for x in X if x < p]

def partitionG(p, X):
    return [x for x in X if x > p]

def elementDeRang(k, X):
    if k >= len(X):
        return None
    pivot = X[0]
    nP = nPetits(pivot, X)
    if k < nP:
        return elementDeRang(k, partitionP(pivot, X))
    if k == nP:
        return pivot
    return elementDeRang(k - nP -1, partitionG(pivot, X))

def medians(X):
     return([elementDeRang(2, X[5*k : 5*k + 5]) for k in
range(len(X)//5)])

def elementDeRangBis(k, X):
    if len(X) < 5:
        return elementDeRang(k, X)
    Y = medians(X)
    pivot = elementDeRangBis(len(Y) // 2, Y)
    nP = nPetits(pivot, X)
    if k < nP:
```

```
        return elementDeRangBis(k, partitionP(pivot, X))
    if k == nP:
        return pivot
    return elementDeRangBis(k - nP - 1, partitionG(pivot, X))


import random as rd
X = [rd.uniform(0, 100) for k in range(8)]
print(X)
print(elementDeRang(2, X))
print(elementDeRangBis(2, X))


from time import clock
X, rang = [rd.uniform(0, 1000) for k in range(10**6 + 2)],
10**5 + 1
# X, rang  = [k for k in range(7*10**2)], 6*10**2

t1 = clock()
u = elementDeRang(rang, X)
t2 = clock()
print('u : ',u, ', durée : ', t2 - t1)

t1 = clock()
u = elementDeRangBis(rang, X)
t2 = clock()
print('u : ',u, ', duréeBis : ', t2 - t1)

# En pratique, elementDeRangBis est beaucoup plus lent, sauf si
?

# Q6 : on montre que   t(n) <= l.n + 3.n.c/4
# d'où c = 4*l convient
```