Suma Divide et Impera

def suma(t,dr,st):  
 #daca subproblema curenta este direct rezolvabila  
 if dr == st:  
 return t[st]  
  
 #etapaDivide  
 mij = (st + dr) // 2  
 sol\_st = suma(t, st, mij)  
 sol\_dr = suma (t, mij + 1, dr)  
  
 #etapaImpera  
 return sol\_st + sol\_dr

Divide et impera combinare solutii

#functie care furnizeaza solutia unei probleme combinand solutiile  
#subproblemelor in care ea a fost descompusa  
def combinare(sol\_st, sol\_dr):  
 pass  
  
def divimp(t, st, dr):  
 #daca subproblema curenta e direct rezolvabila  
 if dr - st <= k #k este, de obicei, 0(un termen) sau 1(2 termeni)  
 return solutie\_problema\_directa  
 else:  
 #etapaDivide  
 mij = (st + dr) // 2  
 sol\_st = divimp(t, st, mij)  
 sol\_dr = divimp(t, mij + 1, dr)  
   
 #etapaImpera  
 return combinare(sol\_st, sol\_dr)

Quickselect(Mergesort)

from random import random.choice  
def quickselect(A, k, f\_pivot=random.choice):  
 pivot = f.pivot(A)  
  
 L = [x for x in A if x < pivot]  
 E = [x for x in A if x == pivot]  
 G = [x for x in A if x > pivot]  
  
 if k < len(L):  
 return quickselect (L, k, pivot)  
 elif k < len(L) + len(E):  
 return E[0]  
 else:  
 return quickselect (G, k- len(L) - len(E), f\_pivot)  
   
#Apel:  
# quickselect(A, k-1)

BFPRT(mediana medianelor) – ! pick.pivot !

def BFPRT(A):  
 if len(A)<=5  
 return sorted(A)[len(A) // 2]  
   
 grupuri = [sorted(A[i:i+5]) for i in range (0,len(A), 5)]  
   
 mediane = [grup(len(grup) // 2) for grup in grupuri]  
   
 return BFPRT(mediane)