TAP C4: Metoda Divide Et Impera

1. Cum calculam eficient min si max dintr-un vector. Propuneți un algoritm care sa facă un număr minim de comparații si care sa garanteze aflarea minimului si maximului. Exista un algoritm care calculează maximul si minimul folosind doar **ceil(3* n/2) - 2** comparații. Este acesta optim? Exista un algoritm divide et impera pentru rezolvarea problemei? Cum calculam complexitatea:

$$T(n) = 2T\left(\frac{n}{2}\right) + 2$$

```
v = [ 7, 9, 1, 6, 5, 6, 11, 3, 10, 4, 8 ]

def minmax(i,j):
    if j - i <= 1:
       return (min(v[i], v[j]) , max(v[i],v[j]))
    else:
        (min_st, max_st) = minmax (i, (i+j)//2 )
        (min_dr, max_dr) = minmax((i + j)//2 + 1, j)
        return ( min(min_st,min_dr), max(max_st, max_dr))

print(minmax(0, len(v) - 1))
```

2. Să se găsească o subsecventă a unui vector de numere întregi cu suma elementelor maximă. Pentru **v** suma maximă este 28 obținută cu elementele subsecventei [15, -1 -10 7 -3 20]. Care este complexitatea algoritmului propus? Este acest algoritm optim?

```
v = [-8, 3, -7, 15, -1, -10, 7, -3, 20, -1, -6, 4]

def submax(i, j):
    if i == j:
        return v[i]
    else:
        max_st = submax(i, (i+j)//2)
        max_dr = submax((i+j)//2 + 1, j)
        max_stm, max_drm, s = 0, 0, v[(i+j)//2]
        for k in range((i+j)//2 - 1, 0, -1):
            max_stm += v[k]
            if max_stm > s:
                s = max_stm
        for k in range((i+j)//2 + 1, j + 1):
            max_drm += v[k]
            if max_drm > s:
                s = max_drm
        return max(max_st, max_dr, s)
```

```
if __name__ == '__main__':
    print(submax(0, len(v) - 1))
```

3. Să se obțină mediana vectorului obținut prin interclasarea a doi vectori a și b de dimensiuni egale:

```
from heapq import merge

u = [14, 17, 20, 22, 30]
v = [12, 31, 40, 42, 45]

def mediana(a, b):
    n = len(a)
    if n == 1:
        return (a[0] + b[0])/2
    if n == 2:
        c = list(merge(a,b))
        return (c[1] + c[2])/2

    if n%2 == 1:
        ma, mb = a[n // 2], b[n // 2]
    else:
        ma = (a[n // 2 - 1] + a[n // 2])/2
        mb = (b[n // 2 - 1] + b[n // 2])/2

    if ma == mb:
        return ma
    if ma > mb:
        return mediana(a[0:n//2 + 1],b[(n-1)//2:n])
    else:
        return mediana(a[(n-1)//2:n],b[0:n//2+1])

if __name__ == '__main__':
    print(mediana(u,v))
```

4. Să se afișeze numărul de inversiuni dintr-un vector. Pentru v = [6, 10, 3, 2, 9, 7, 1, 4, 8, 5] se va afișa 25.

```
v = [6, 10, 3, 2, 9, 7, 1, 4, 8, 5]

def inv(st, dr):
    if st == dr:
        return 0
    s = []                  #v[st:dr+1] sortat
    m = (st + dr)//2
    nrs = inv(st, m)
    nrd = inv(m + 1, dr)
    vs = v[st:m+1]          #subvectorul stang (2, 3, 6, 9, 10)
    vd = v[m+1:dr+1]        #subvectorul stang (1, 4, 5, 7, 8)
    nrm = 0                 #numarul de inv (a,b) a din vs, b din vd
    i, j = 0, 0             #se interclaseaza in s vectorii vs si vd
    while i < len(vs) and j < len(vd):
        if vs[i] < vd[j]:
            s.append(vs[i])
            nrm += j        #nr de valori din vd mai mici dect vs[i]
            i += 1
        else:
            s.append(vd[j])
            j += 1
    while i < len(vs):
        s.append(vs[i])
        nrm += j
        i += 1
    while j < len(vd):
        s.append(vd[j])
        j += 1
    v[st:dr+1] = s
    return nrs + nrd + nrm

if __name__ == '__main__':
    print(inv(0, len(v)-1))
```

5. Aplicați teorema master pentru recurentele:

a) $T(n) = 2T\left(\frac{n}{2}\right) + n^2$

b) $T(n) = 8T\left(\frac{n}{2}\right) + 12n^2$

c) $T(n) = 2T\left(\frac{n}{2}\right) + 13n$

d) $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$

e) $T(n) = 3T\left(\frac{n}{3}\right) + \sqrt{n}$