1) Se dau n cuburi cu laturile diferite două câte două. Fiecare cub are o culoare, codificată cu un număr de la 1 la p (p dat).

a) Să se construiască un turn de înălțime maximă de cuburi în care un cub nu poate fi așezat pe un cub de aceeași culoare sau cu latură mai mică decât a sa – O(n logn). Pe prima linie a fișierului de intrare se dau n şi p, iar pe următoarele linii latura şi culoarea fiecărui cub. În fișierul de ieșire se vor afișa înălțimea turnului obținut şi indicele cuburilor folosite (de la bază la vârf)

date.in	date.out
4 2	23
5 1	2 4 1
10 1	
9 1	
8 2	

b) În cazul în care lungimile laturilor cuburilor nu erau diferite mai este valabilă metoda propusă? Justificați.

```python
fin = open("data.in","r")
fout = open("data.out","w")
l = [int(x) for x in fin.readline().split()]
n, p = l[0],l[1]
print(n,p)
cuburi = []

for i in range(n):
    l = [int(x) for x in fin.readline().split()]
    cuburi.append((l[0],l[1],i))

cuburi.sort(key = lambda x: -x[0])
print(cuburi)

sol = []
height = 0
sol.append(cuburi[0][2])
height += cuburi[0][0]
lastCol = cuburi[0][1]

for i in range(1,n):

    if cuburi[i][1] != lastCol:
        sol.append(cuburi[i][2])
        lastCol = cuburi[i][1]
        height += cuburi[i][0]

print(height,file=fout)

for el in sol:
    print(el+1,end=' ',file=fout)

```

2) Avem la dispozitie un interval inchis [A,B] si o multime de alte N intervale inchise [Ai,Bi], $1 \leq i \leq N$. Scrieti un program care calculeaza si afiseaza numarul minim de intervale inchise din multimea data cu proprietatea ca prin reuniunea acestora se obtine un interval care include pe [A,B].

acoperire.in	acoperire.out
13 28 9 1 12 3 20 15 19 25 34 17 23 24 25 13 20 11 16 23 27	4

```python
fin = open("data.in","r")
fout = open("data.out","w")
l = [int(x) for x in fin.readline().split()]
a, b = l[0],l[1]
n = int(fin.readline())
print(a,b,n)
intervale = []
for i in range(n):
    l = [int(x) for x in fin.readline().split()]
    intervale.append((l[0],l[1]))

intervale.sort(key = lambda x: x[0])
print(intervale)
r = -1
sol = 0
for interval in intervale:

    if a>b:
        break
    if interval[0] <= a:
        r = max(r,interval[1])
    else:
        sol += 1
        a = r
        if interval[0] <= a:
            r = max(r,interval[1])

sol += 1
print(sol)

```

Divide et Impera

3)Se dă un vector a=(a1,…an) de tip munte (există un indice i astfel încât a1<a2<…<ai > ai+1>…>an;

ai se numește vârful muntelui). Propuneți un algoritm O(log n) care determină vârful muntelui (în

calculul complexității algoritmului nu se consideră și citirea vectorului).

Intrare:

5

4 8 10 11 5

Raspuns:

11

```
f = open("input.txt", "r")
n = int(f.readline())
v = [int(x) for x in f.readline().split()]
v.append(0) #element sa un faca urat linia 76 (mij + 1)
st = 0
dr = n - 1
varf = 0
while st <= dr:
    mij = (st + dr) // 2
    if (mij == 0 and v[mij] > v[mij + 1]) or (mij == n - 1 and v[mij] > v[mij - 1]):
        varf = v[mij]
        st = dr + 1
    elif v[mij] > v[mij - 1] and v[mij] > v[mij + 1]:
        varf = v[mij]
        st = dr + 1
    elif v[mij] > v[mij - 1]:
        st = mij + 1
    else:
        dr = mij - 1
print(varf)
f.close()
```