Prvé domáce kolo 19. sezóny súťaže Palma sa konalo dňa `28.01.2021` v čase `15:00-18:00`. Zúčastnil sa ho rekordný počet 62 súťažných tímov (celkovo 105 súťažiacich) z rôznych kútov Slovenska (a dokonca dva tímy z Česka), ktorí riešili 4 úlohy, pričom dve z nich boli rozdelené na ľahšiu a ťažšiu verziu. Víťazný tím zvládol vyriešiť všetky úlohy už za necelých 35 minút.

Podľa úspešne odovzdaných riešení vo výsledkovej listine najjednoduchšou úlohou bola ľahká verzia prvej úlohy (vyriešená 37 tímami) a najobtiažnejšou ťažšia veria tejto úlohy, ktorú vyriešilo iba 5 tímov.

A Kĺzavý priemer

Úloha Kĺzavý priemer mala priamočiare riešenie - vygenerovať všetky K-tice, každú usporiadať a vypísať z nej medián. Pri použití triviálneho usporiadania sa takto dosiahne časová zložitosť $O(N \cdot K^2)$. Použitím efektívnejšieho usporiadania dostávame zložitosť $O(N \cdot K \cdot \log K)$. Zápis v jazyku Python 3 by mohol vyzerať nasledovne:

```
s = int(input()) #pocet testovacich sad
for i in range(s): #pre kazdu testovaciu sadu, i-pocitadlo
        #nacitanie medzerou oddelenych cisel
        n, k = [int(_) for _ in input().split()]
        pole = [int(_) for _ in range(n)] #nacitanie pola n cisel
        for zac in range(n-k+1): #pre vsetky zaciatky k-tic
                okno = pole[zac:zac+k] #k hodnot z pola od indexu zac
                usp = sorted(okno) #usporiadane okno
                #median je prostredny prvok
                #v pripade neparneho poctu lavy z dvojice v strede
                print(usp[(k-1) // 2]) #celociselne delenie
```

S využitím myšlienky usporiadania vsúvaním (`InsertSort`) je možné pamätať si ostatných K hodnôt a stále ďalšie vkladať do usporiadanej postupnosti, čo umožní dosiahnuť zložitosť $O(N \cdot K)$.

Na riešenie ťažšej verzie úlohy je potrebné riešenie s časovou zložitosťou $O(N \cdot \log K)$, čo vyžaduje použitie dátovej štruktúry s vkladaním, odstránením hodnoty ako aj nájdenia mediánu v logaritmickom čase. Naše vzorové riešenie pracovalo s rozdelením okna na tri časti - menšie (halda s maximom v koreni), medián a väčšie prvky (halda s minimom v koreni). Iné riešenia obsahovali dátové štruktúry a metódy: `multiset.lower_bound`, `ordered_set.find_by_order`, `tree.find_by_order`, `treap.split_by_count`.

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

B Karanténa

Úloha Karanténa bola priamočiarym uplatnením prehľadávania grafu (v ktorom vrcholy reprezentujú osoby a hrany kontakty medzi nimi) s časovou zložitosťou $O(N + M)$, pričom najjednoduchšie je spustiť prehľadávanie zo všetkých pozitívne testovaných osôb. Možné riešenie v jazyku Python 3:

```
#nacitanie poctu osob, pozitivnych a kontaktov
n, p, m = [int(_) for _ in input().split()]
#nacitanie pozitivne testovanych osob
pozit = [int(_) for _ in input().split()]
#nacitanie kontaktov
kontakt = [[] for _ in range(n+2)] #pole kontakov osoby
for _ in range(m): #m kontaktov
  #nacitanie jedneho kontaktu
  i, j = [int(_) for _ in input().split()]
  #kontakty su obojstranne
  kontakt[i].append(j)
  kontakt[j].append(i)
#pole bol oznacuje uz spracovane osoby
#na zaciatku su to vsetky pozitivne testovane osoby
bol = [True for i in range(n+2) if i in pozit else False]
pocet = len(pozit)
#pole/zasobnik osob na prehladavanie
#na zaciatku vsetky pozitivne testovane osoby
q = pozit
while len(q) > 0:
  #spracujeme dalsiu osobu zo zasobnika
  #teda prehladame vsetky jej kontakty
  v = q.pop()
  for w in kontakt[v]: #vsetky kontakty osoby v
    if not bol[w]: #ak este kontakt nebol spracovany
      bol[w] = True #oznacenie ako spracovany
      pocet += 1 #zvysenie poctu osob v karantene
      q.append(w) #pridanie na dalsie spracovanie
print(pocet) #vypisanie vysledneho poctu
```


Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

C Deravá doska

Riešením úlohy Deravá doska je hľadanie pohybu figúrky na hracej ploche. Najvýhodnejšie riešenie je vytvorenie si zoznamu pohybov (ktorý sa len upraví podľa použitej figúrky) a skúšanie všetkých možných pohybov - či sa nachádzajú v hracej ploche a či je dané políčko voľné.

```
#skoky jazdca
skoky = [(1, 2), (2, 1), (-1, 2), (-2, 1),
         (1,-2), (2,-1), (-1,-2), (-2,-1)]
s = int(input()) #pocet testovacich sad
for _ in range(s): #pre kazdu sadu
  #nacitanie rozmerov dosky
  m, n = [int(_) for _ in input().split()]
  #nacitanie dosky
  doska = [input() for _ in range(m)]
  #nacitanie pozicie jazdca
  r, s = [int(_) for _ in input().split()]
  #precislovanie, kedze Python indexuje od nuly
  r -= 1; s -= 1
  pocet = 0
  for skok in skoky: #vsetky skoky
    #nova pozicia po skoku
    n_r, n_s = r+skok[0], s+skok[1]
    #overenie, ci je v ramci hracej plochy
    if 0 <= n_r < m and 0 <= n_s < n:
      #overenie ci na danej pozicii nie je diera
      if doska[n_r][n_s] == "1":
        pocet += 1 #zvysime pocet moznosti
  print(pocet)
```


Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

D Doska

Ľahšia verzia úlohy Doska sa dá riešiť triviálne, a to hľadaním minimálnej vzdialenosti medzi všetkými dvojicami vrcholov v časovej zložitosti $O(N^2)$. Namiesto priameho počítania druhej odmocniny zo súčtu druhých mocnín je výhodnejšie použiť funkciu `math.hypot`.

```
import math #pre vypocet vzdialenosti

n=int(input()) #nacitanie poctu bodov
#nacitanie bodov
body = []
for _ in range(n):
  #nacitanie suradnic bodu
  x, y = [int(_) for _ in input().split()]
  body.append((x, y))
#pocitanie najmensej vzdialenosti
#zacneme dostatocne velkou vzdialenostou
vzd = 10**9
#pre vsetky dvojice bodov i < j
for i in range(n-1):
  for j in range(i+1, n):
    #vzdialenost dvoch bodov
    vzd = min(vzd, math.hypot(body[i][0]-body[j][0], body[i][1]-body[j][1]))
print(vzd)
```

Riešenie je možné zrýchliť (aj keď s rovnakou časovou zložitosťou) s využitím `itertools.combinations`.

```
import itertools
import math
n = int(input()) #pocet bodov
#nacitanie suradnic
body = [tuple(map(int, input().split())) for _ in range(n)]
#najdenie a vypisanie minimalnej vzdialenosti dvojic bodov
print(min([math.hypot(a[0]-b[0], a[1]-b[1])
           for a, b in itertools.combinations(body, 2)]))
```

Riešenie ťažšej verzie úlohy Doska je jednou z typických úloh Zametania (`Sweep Line`). Body sa usporiadajú podľa súradníc a zametajú sa zľava doprava (podľa tzv. zametacej priamky). Priebežne sa ukladá doteraz najmenšia vzdialenosť a zoznam bodov, ktoré sú podľa x-ovej súradnice vo vzdialenosti menšej ako doterajšie minimum. Takéto riešenie usporiadaného zoznamu je možné v lineárnom čase, celková časová zložitosť je $O(N \cdot \log N)$. Odovzdané riešenia obsahovali rekurzívne binárne rozdelenie podľa mediánu bodov (na štýl usporiadania spájaním - `MergeSort`) či použitie dátových štruktúr a metód: `set.lower_bound`.

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja
OPERAČNÝ PROGRAM ĽUDSKÉ ZDROJE
Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje