Univerzitetni programerski maraton

FINALE 2018 – rešitve nalog

Tomaž Hočevar

</upm UNIVERZITETNI PROGRAMERSKI MARATON

Plezalne smeri (16/16, muzik.si – 0:12)

Uredi smeri po njihovih težavnostih.

```
nemska IV-
bavarska IV+
slovenska III
sfinga VII
ljubljanska VII-
```

```
slovenska
nemska
bavarska
ljubljanska
sfinga
```

- rimske številke
  - seznam, slovar, …
  - seznam s “+” in “-”

```
bavarska IV+
(4, +1, 'bavarska')
```

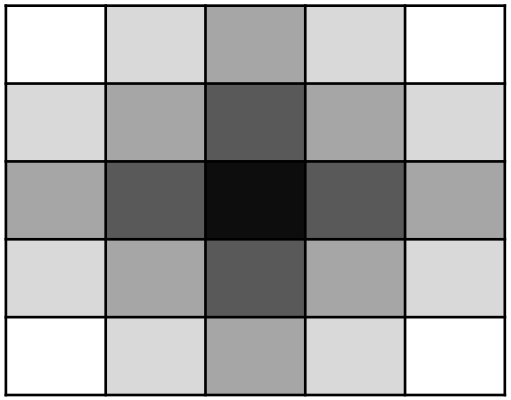
- urejanje po več kriterijih
  - terke, primerjalna funkcija, …

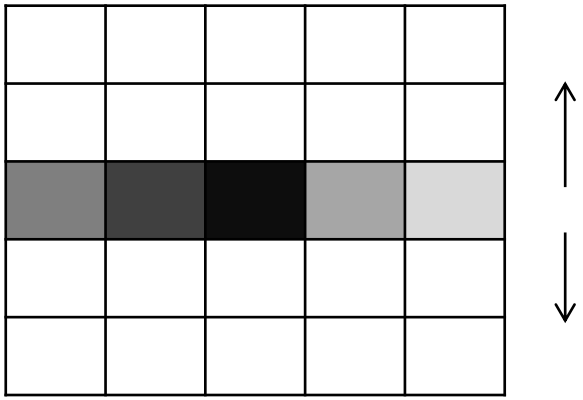
Plezalne smeri

```
n = int(input())
rim = ["I","II","III","IV","V","VI","VII","VIII","IX","X","XI","XII"]
smeri = [x for r in rim for x in [r+"-", r, r+"+"]]
s = []
for i in range(n):
    name, grade = input().split(" ")
    s.append((smeri.index(grade), name))
s.sort()
for _, x in s:
    print(x)
```

Kvadrat števil (16/16, ASMx64 – 0:19)

Razvrsti elemente v celice kvadratne mreže tako, da bodo po velikosti padali od sredine proti robovom.





4	12	20	5	1
11	19	24	13	6
18	23	25	21	14
10	17	22	15	7
3	9	16	8	2

13	14	15	12	11
18	19	20	17	16
23	24	25	22	21
8	9	10	7	6
3	4	5	2	1

</upm UNIVERZITETNI PROGRAMERSKI MARATON

Kvadrat števil

```
n = int(input())
val = sorted([int(input()) for i in range(n*n)], reverse=True)
pos = sorted([(abs(n//2-x)+abs(n//2-y), y, x)
              for y in range(n) for x in range(n)])
t = [[0]*n for i in range(n)]
for i in range(n*n):
    t[pos[i][1]][pos[i][2]] = str(val[i])
for i in range(n):
    print(' '.join(t[i]))
```

```
n = int(input())
val = sorted([int(input()) for i in range(n*n)])
t = [[0]*n for i in range(n)]
for y in list(range(n//2,-1,-1)) + list(range(n//2+1,n,+1)):
    for x in list(range(n//2,-1,-1)) + list(range(n//2+1,n,+1)):
        t[y][x] = str(val.pop())
for i in range(n):
    print(' '.join(t[i]))
```

</upm UNIVERZITETNI PROGRAMERSKI MARATON

Tabela (15/16, jam – 0:14)

Poravnaj stolpce podane tabele.

```
LRL
Miha,10000,EUR
Aleksander,10,BTC
```

```
Miha       | 10000 | EUR
Aleksander |    10 | BTC
```

- poznavanje programskega jezika
  - branje podatkov (CSV)
  - najdaljša beseda v vsakem stolpcu
  - izpis (poravnava)

Tabela

```
def align(caw):
    return ('{:' + caw[1] + str(caw[2]) + '}').format(caw[0])

r, c = map(int, input().split())
col_align = input().replace('L','<').replace('R','>')
data = [input().split(',') for i in range(r)]
col_width = [max(len(row[i]) for row in data) for i in range(c)]
for row in data:
    print(' | '.join(map(align, zip(row, col_align, col_width))))
```

Fruit ninja (10/15, Final Solution – 0:47)

Preštej, koliko krogov ima presek s podano daljico.

- analitična rešitev
  - krajišče v krogu ali presek kroga z daljico

$ax + by = c$ $\qquad\qquad$ $(x-p)^2 + (y-q)^2 = r^2$

$ax_1 + by_1 = c, \qquad ax_2 + by_2 = c$

$a(x_1 - x_2) = b(y_2 - y_1)$

$a = y_2 - y_1 \quad b = x_1 - x_2 \quad c = ax_1+by_1$

$a \neq 0$ ... swap(x,y)

$Ay^2 + By + C = 0$

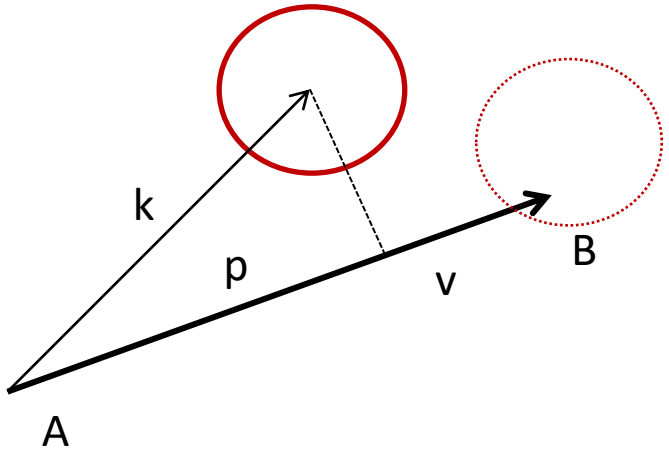
</upm UNIVERZITETNI PROGRAMERSKI MARATON

Fruit ninja

- geometrijska rešitev
  - projekcija središča na nosilko

$|p| = k \cdot v / |v|$

$p = |p| \cdot (v / |v|) = (k \cdot v) / |v|^2 \cdot v = ((k \cdot v) / (v \cdot v)) \cdot v$



- natančnost

  r += EPS

Fruit ninja

```
x1, y1, x2, y2 = map(int, input().split())
vx, vy = x2-x1, y2-y1
n = int(input())
st = 0
for i in range(n):
    p, q, r = map(int, input().split())
    r += 1e-4
    if (p-x1)**2+(q-y1)**2<=r**2 or (p-x2)**2+(q-y2)**2<=r**2:
        st += 1
    else:
        kx, ky = p-x1, q-y1
        a = (kx*vx+ky*vy)/(vx*vx+vy*vy)
        if a < 0 or a > 1: continue
        x, y = x1+a*vx, y1+a*vy
        if (p-x)**2+(q-y)**2<=r**2:
            st += 1
print(st)
```

Arbitraža (9/12, Final Solution – 1:46)

Izberi in uredi argumente, ki maksimizirajo kriterijsko funk.

- opažanja
  - "uganemo" a in b
  - izberemo najmočnejše arg.
  - najmočnejše arg. množimo med seboj ($p_1 \cdot s_1 + \ldots$)
    - kaj če p_1 ne množimo z s_1?

p_1	p_2	p_3	p_a	p_5	p_n
s_1	s_2	s_b	s_m		

$\max(a,b) \cdot (p_1 s_1 + p_2 s_2 + p_3 s_b + p_a)$

$p_1 s_1 + p's' >= p_1 s' + p' s_1$

$(p_1 - p')(s_1 - s') >= 0$

</upm UNIVERZITETNI PROGRAMERSKI MARATON

Arbitraža

- preverimo vse pare a, b
  - $b = k - a$
  - $ps(i) = \sum_{j=1..i} p(j) \cdot s(j)$
  - $p(i) = \sum_{j=1..i} p(j)$
  - $s(i) = \sum_{j=1..i} s(j)$

p_1	p_2	p_3	p_a	p_5	p_n
s_1	s_2	s_b	s_m		

$$\max(a,b) \cdot (p_1 s_1 + p_2 s_2 + p_3 s_b + p_a)$$

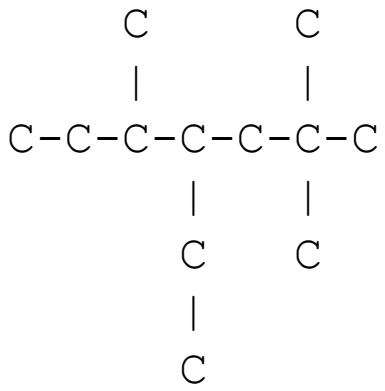
Arbitraža

```
n, m, k = map(int,input().split())
p = sorted(list(map(int,input().split())), reverse=True)
s = sorted(list(map(int,input().split())), reverse=True)

sum_p, sum_s, sum_ps = [0], [0], [0]
for i in range(n): sum_p.append(sum_p[-1] + p[i])
for i in range(m): sum_s.append(sum_s[-1] + s[i])
for i in range(min(n,m)): sum_ps.append(sum_ps[-1] + p[i]*s[i])

sol = 0
for a in range(min(k,n)+1):
    b = min(k-a, m)
    if a >= b: r = a * (sum_ps[b] + (sum_p[a] - sum_p[b]))
    else: r = b * (sum_ps[a] + (sum_s[b] - sum_s[a]))
    sol = max(sol, r)
print(sol)
```

Alkani (?/4, Final Solution – 3:53)

Poimenuj narisane alkane (z “enostavnimi” radikali).

- naredi, kar piše (implementacija)
  - poišči komponente/spojine na sliki
  - poišči najdaljšo pot v drevesu
  - določi dolžine in pozicije radikalov
  - izberi številčenje z leve oz. desne
  - združi/preštej enako velike radikale
  - oblikuj ime alkana (vrstni red radikalov in pozicije)
  - združi/preštej imena vseh alkanov



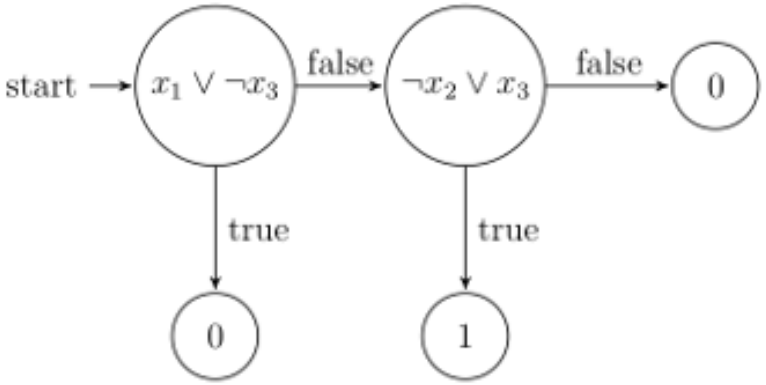
4-etil-2,2,5-trimetil-heptan

</upm UNIVERZITETNI PROGRAMERSKI MARATON

Odločitveni seznam (0/0)

Sestavi čim krajši odločitveni seznam, ki ustreza podanim vhodnim in izhodnim logičnim vrednostim.

- 2^{2n} disjunkcij v različnih vrstnih redih?
- vsaka spremenljivka nastopa največ enkrat
  - x_3 in $!x_3$ je zavajujoč
- obstaja rešitev z 1 spremenljivko v pogojih



1 0 1 --> 0
0 1 0 --> 0
0 0 1 --> 1
0 1 1 --> 1

</upm UNIVERZITETNI PROGRAMERSKI MARATON

Odločitveni seznam

- vrednosti izhodov alternirajo, recimo da je prvi enak 1
- zberemo v disjunkcijo vse spremenljivke (ali negacije), ki imajo željen izhod (požrešna strategija)

x1

1 1 1 1 1 --> 1
1 0 1 1 0 --> 1
0 1 1 0 1 --> 1
0 1 0 0 1 --> 0
0 0 0 1 0 --> 0
1 0 0 0 0 --> 1
0 1 1 1 1 --> 0
1 0 0 1 0 --> 1
0 0 0 1 1 --> 0
1 1 1 1 0 --> 1
0 0 1 1 0 --> 0
0 1 0 1 0 --> 0
0 0 0 0 0 --> 0
1 1 0 0 0 --> 1
1 1 1 0 0 --> 1

!x2 ali !x3 ali x4

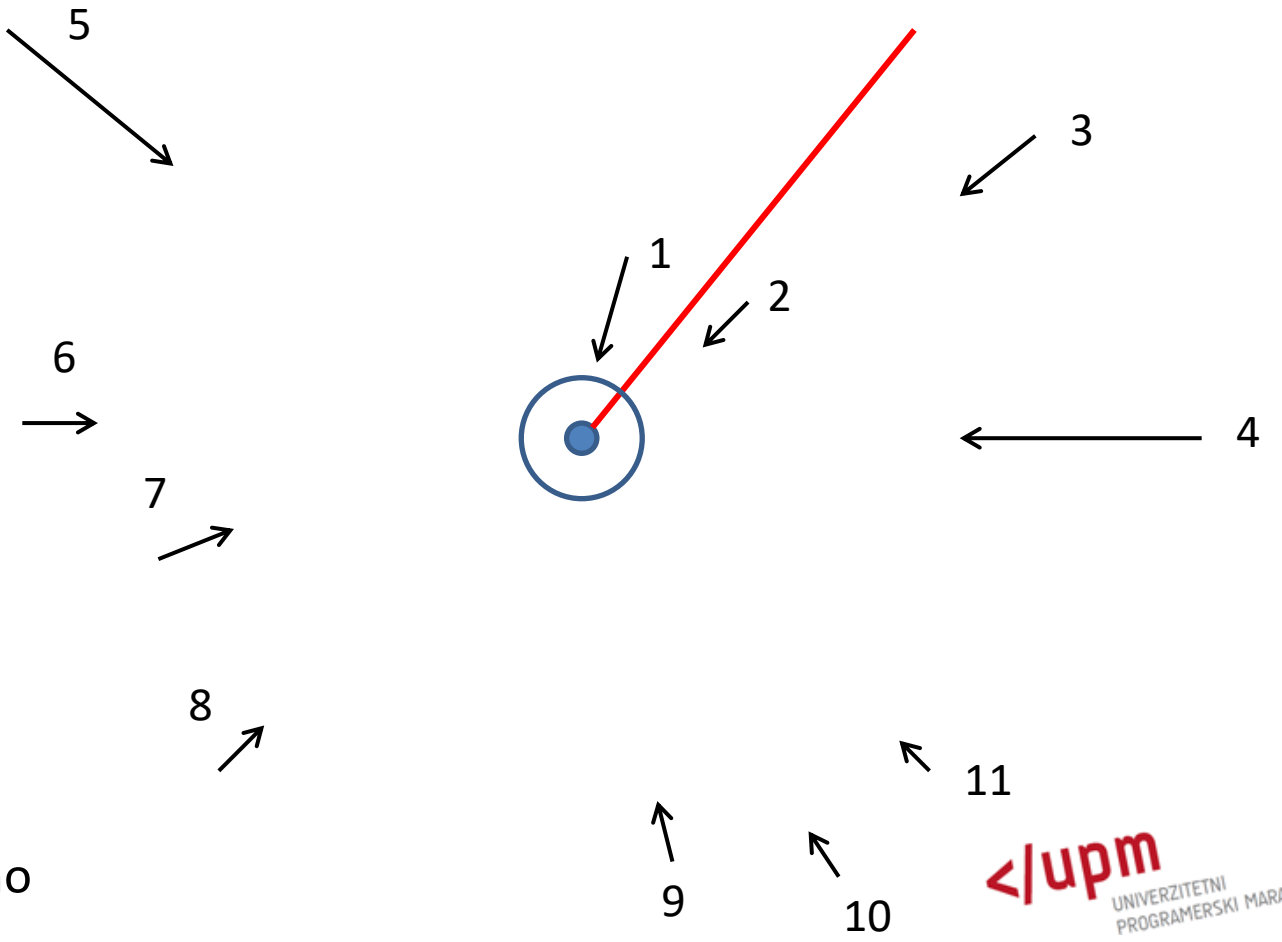
0 1 1 0 1 --> 1
0 1 0 0 1 --> 0
0 0 0 1 0 --> 0
0 1 1 1 1 --> 0
0 0 0 1 1 --> 0
0 0 1 1 0 --> 0
0 1 0 1 0 --> 0
0 0 0 0 0 --> 0

1

0 1 1 0 1 --> 1

Gusarji (?/2 Final Solution – 4:30)

Z optimalnim obračanjem laserja obvaruj ladjo pred gusarji.

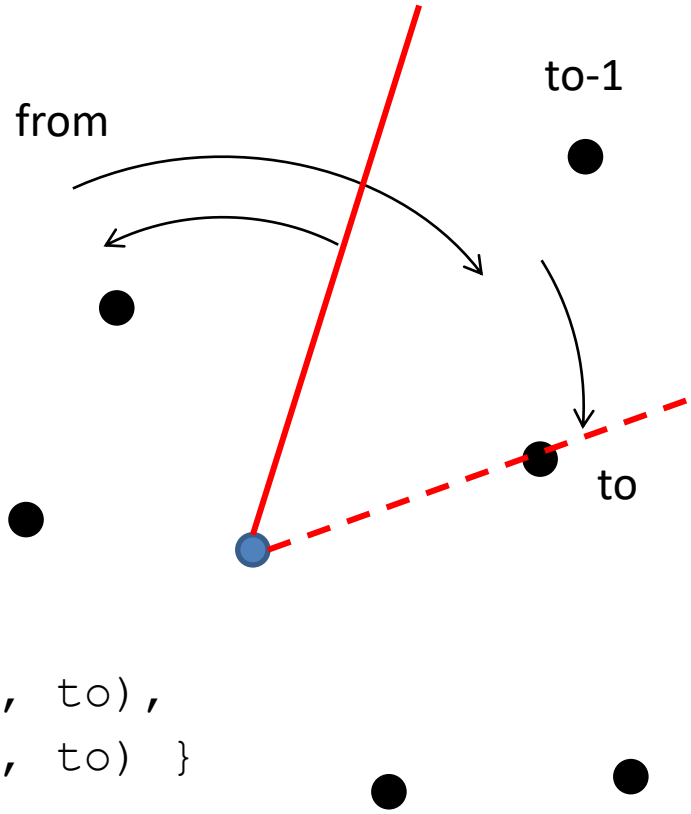


- vrstni red?
  - hitri
  - bližnji
  - levo oz. desno

Gusarji

- uredimo gusarje po kotu
  - uničeno bo strnjeno podzaporedje
- dinamično programiranje
  - podproblem (from, to, now)

    now: 0=from, 1=to

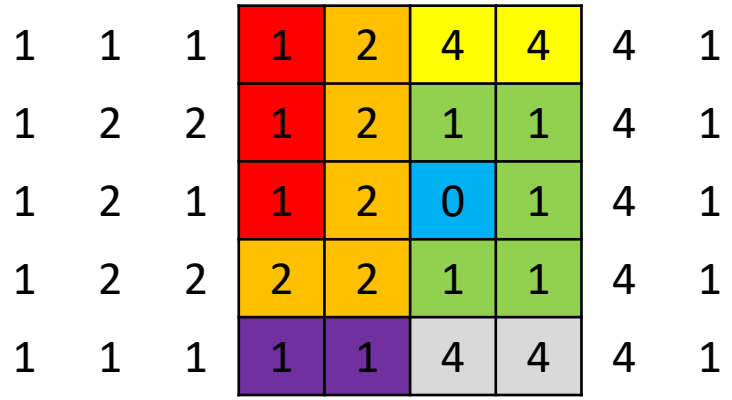



```
f(from, to, 1) =
  min{ f(from, to-1, 1) + rot(to-1, to),
       f(from, to-1, 0) + rot(from, to) }
```

  - $O(n^2)$
  - nadloge: različne hitrosti, koti, krožen seznam, varnostna razdalja, …

Zemljiški kataster (?/4)

Preštej število povezanih komponent na določenih območjih.

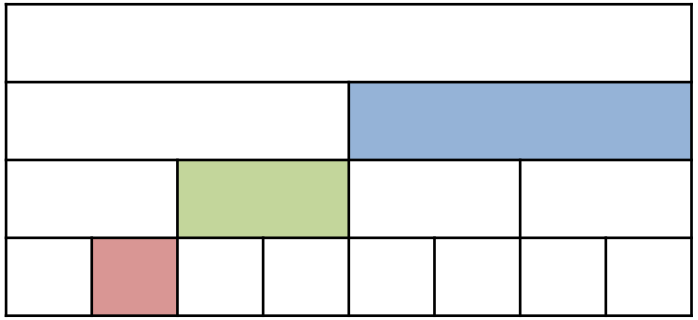


Omejitve
- največja tabela: 100 x 10 000
- število poizvedb: 100 000

</upm UNIVERZITETNI PROGRAMERSKI MARATON

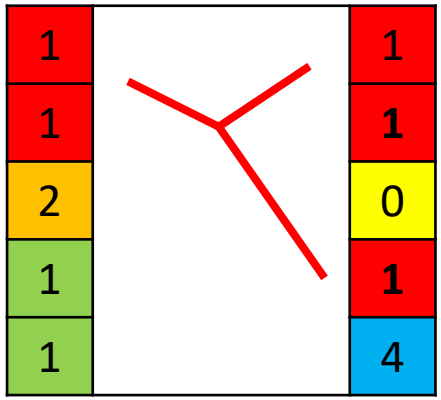
Zemljiški kataster

- segment/range tree
  - poizvedba: O(x log n)
  - izgradnja: O(x n)
- vsebina vozlišča
  - število komponent
  - še kaj?
- združevanje
  - levi in desni “profil”
  - O(x)

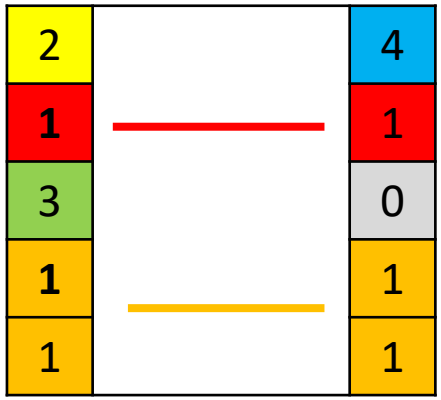


1	1	1	1	2	4	4	4
1	2	2	1	2	1	1	4
1	2	1	1	2	0	1	4
1	2	2	2	2	1	1	4
1	1	1	1	1	4	4	4

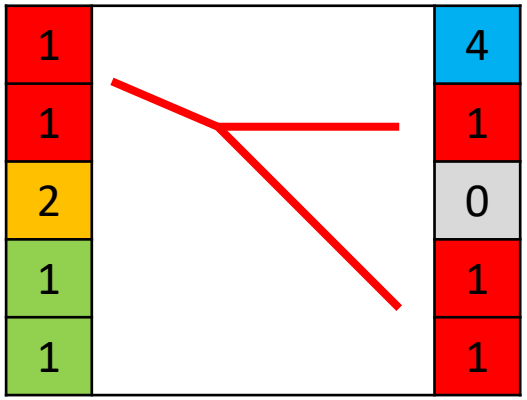
Zemljiški kataster



+



=



{A1, A2, B1, **B2**, **B4**}
{A3} {A4, A5} {B3} {B5}

{C1} {**C2**, D2} {C3}
{**C4**, C5, D4, D5} {D1} {D3}

{A1, A2, D2, D4, D5} {A3}
{A4, A5} {D1} {D3}

- združevanje
  - disjunktne množice (disjoint-set, union-find)
- O(q log(n) m)

</upm UNIVERZITETNI PROGRAMERSKI MARATON

Zaključek

- lažje naloge kot prejšnja leta
  - višje povprečje točk
  - podoben maksimum
- hitrost, znanje
- strategija izbora nalog
  - pričakovan čas reševanje
  - pričakovana pravilnost rešitve