# Univerzitetni programerski maraton

## FINALE 2024 – rešitve nalog

**Tomaž Hočevar**

</upm UNIVERZITETNI PROGRAMERSKI MARATON

# Račke

Preštej pojavitve podanega vzorca račke na sliki.

- preverimo ujemanje vzorca na vseh možnih mestih
  - dodamo prazno obrobo
  - velikost slike ≤ 10 000, velikost vzorca ≤ 100

```
........#...........
.##..####..#........
###.....#.#....##....
..####...#.....###...
..###....#..####.....
.........#...###.....
...##.....#....  ##...
...###..........###..
####.........####....
.###.##.......###.##.
.....###.............
..####........##.....
...###........###....
```

```
*....***
..##.***
.###....
...####.
**.###..
**.....*
```

```
***....*
***.##..
....###.
.####...
..###.**
*.....**
```

```
rh, rw = len(racka), len(racka[0])
racke = [racka, [row[::-1] for row in racka]]

grid = ["."+row.strip()+"." for row in sys.stdin]
w = len(grid[0])
grid = ["."*w] + grid + ["."*w]
h = len(grid)

st = 0
for y in range(h-rh+1):
    for x in range(w-rw+1):
        for d,racka in enumerate(racke):
            if all(grid[y + i][x + j] == racka[i][j]
                   for i in range(rh) for j in range(rw)
                   if racka[i][j] != "*"):
                st += 1
print(st)
```

# D'Hondtova metoda

Izračunajte rezultate volitev po D'Hondtovi metodi.

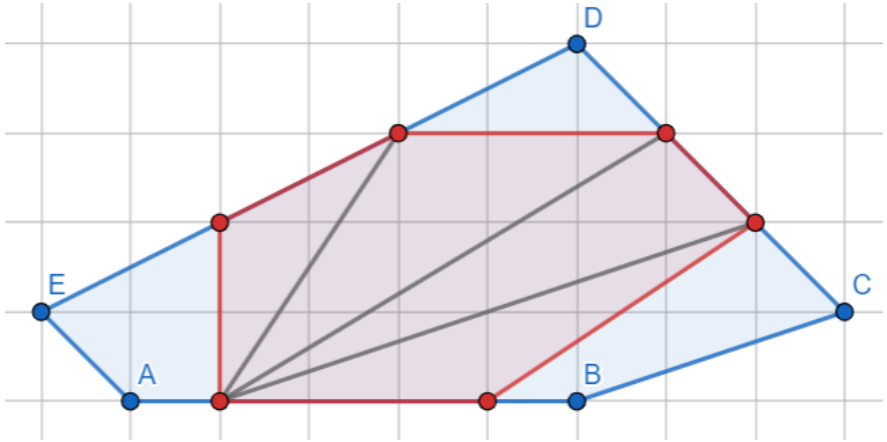
- Kako razdeliti M sedežev med N strank glede na glasove?
- uredimo kandidate znotraj stranke
  - najprej glede na doseganje praga preferenčnih glasov
    - preferenčne glede na število glasov
    - ostale glede na pozicijo na listi
- uredimo vseh N*M količnikov s kandidati, ki jim ustrezajo
  - enake glede na število glasov stranke
- pozor: ponovljena imena, enako število glasov

```python
def read():
    name, votes = input().rsplit(' ', 1)
    return name, int(votes)

m, n = map(int, input().split())
all_candidates = []
for i in range(n):
    party, party_votes = read()
    candidates, all_votes = [], 0
    for j in range(m):
        name, votes = read()
        all_votes += votes
        candidates.append(
            ((-votes if votes * 2 * m >= party_votes else 0), j, name)
        )
    all_candidates += [
        (party_votes / (1 + j), party_votes, f"{name} ({party})")
        for j, (_, _, name) in enumerate(sorted(candidates))
    ]

for _, _, name in sorted(all_candidates, reverse=True)[:m]:
    print(name)
```

# Nenavadna parcela

Poišči večkotnik z največjo ploščino s celoštevilskimi oglišči na robovih danega konveksnega večkotnika.

- izberemo vse celoštevilske točke na robu
  - dovolj sta skrajni dve
  - $x, y = x_1 + dx/\gcd(dx,dy), y_1 + dy/\gcd(dx,dy)$
- izračunamo površino
  - po trikotnikih z vektorskim produktom



# Pločevinke v pravokotniku

Iz P pločevink formiraj pravokotnik s čim manjšim obsegom.

- veljavni pravokotniki: P = a*b
- za vsak test preverimo vse delitelje P-ja kot možne dimenzije
  - min 2*(a+b)-4, da velja a*b = P (pazi a=1)
  - TLE: O(T √P)
- izračunamo rezultate s “sitom”, kjer označujemo večkratnike
  - O(P log(P))

```
for (int p=1; p<=N; p++) perim[p]=p;
for (int a=2; a*a<=N; a++) {
    for (int b=a; a*b<=N; b++) {
        int n = a*b;
        perim[n] = min(perim[n], 2*(a+b)-4);
    }
}
```

</upm UNIVERZITETNI PROGRAMERSKI MARATON

# Popravljanje žetonov

Preštej besede (žetone), ki se začnejo s čim daljšo pripono seznama besed (konteksta).

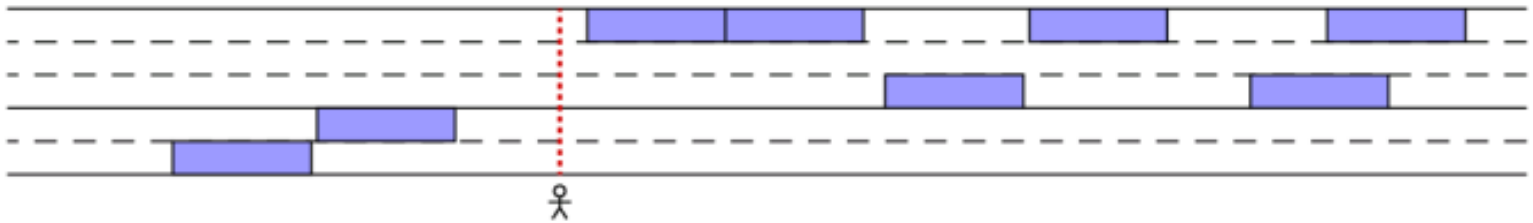
- dolžina žetona ≤ 30
  - relevantnih zadnjih 30 žetonov v kontekstu
  - obravnavamo 30 pripon dolžine max 30 znakov
- organiziramo žetone
  - črkovno drevo (trie)
    - operacij: N*30 + M*(300 + 30*30)
  - urejanje, bisekcija
  - slovar možnih predpon

```
er
ma
maraton
pro
program
ra
ski
ton

4 4 0 6 1 =
... er ski ma
```

</upm UNIVERZITETNI PROGRAMERSKI MARATON

# Prečkanje ceste

Poiščite prvi čas, ko lahko pešec varno prečka cesto.



- smer vožnje ni pomembna
- vsak avto ima svoje “nevarno” časovno obdobje čakanja
  - avto ravno zbije pešca na koncu pasu
  - pešec stopi na pas tik za avtom
- unija obdobij
  - uredimo dogodke (začetke, konce, čas 0)
- računske napake (float)!
  - celoštevilska aritmetika (fiksen imenovalec ulomkov)

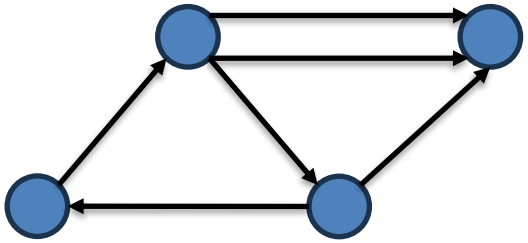
# Velikost C-jevske strukture

Izračunajte velikost veljavne C-jevske structure.

- normaliziramo whitespace
- izračunamo pare oklepajev
- razčlenjevanje izraza (velikost, največji tip)
  - var -> velikost
    `bam[1][2][1]`
  - vars -> velikost
    `bim, bam[1][2][1], bom[2]`
  - struct -> največji, velikost (večkratnik največjega)
    `struct { bool prvi; int drugi[3][2], tretji; }`
  - declaration -> največji, velikost
    `int bim, bam[1][2][1], bom[2]`
  - declarations -> največji, velikost (poravnava posam.)
    `int a, b; char c; short neki[3][4];`

# Zamude

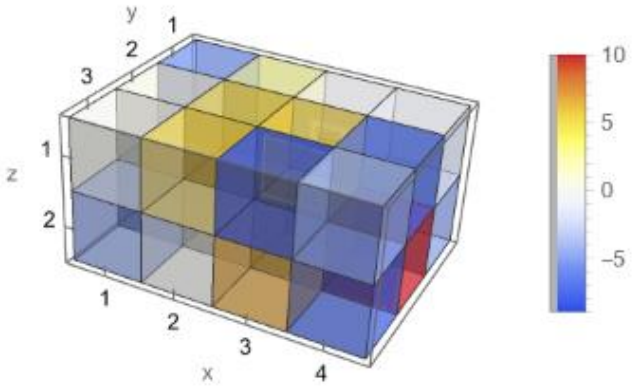
Izračunaj najmanjši pričakovan prihod z letališča 1 na N ob optimalnem izbiranju letov v primeru zamud.



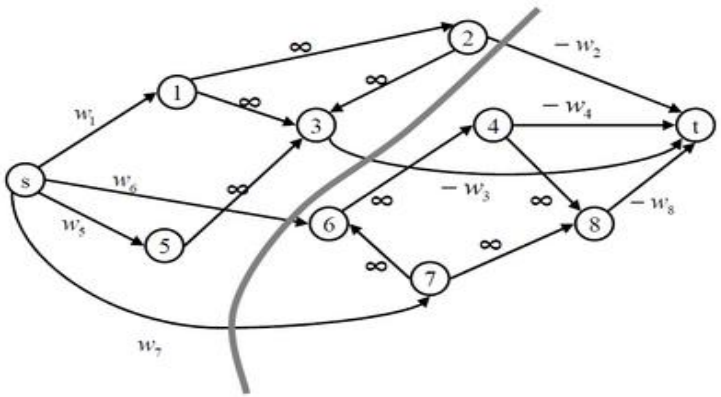
- f(x, t) = pričakovan čas potovanja, če smo na x ob času t
  - relevantni t-ji so časi odhodov letov
  - za izračun f(x, t) potrebujemo rezultate za t' > t
- računamo rezultate po padajočih t-jih
  - obravnavamo lete po padajočih odhodih
  - za vsak x hranimo seznam rezultatov za že izračunane t-je
    - map ali bisekcija po urejenem seznamu
  - utežena vsota leta brez zamude in z zamudo

# Rudnik

Izračunajte izkop s čim večjo vrednostjo ob upoštevanju omejitev za zagotavljanje stabilnosti.

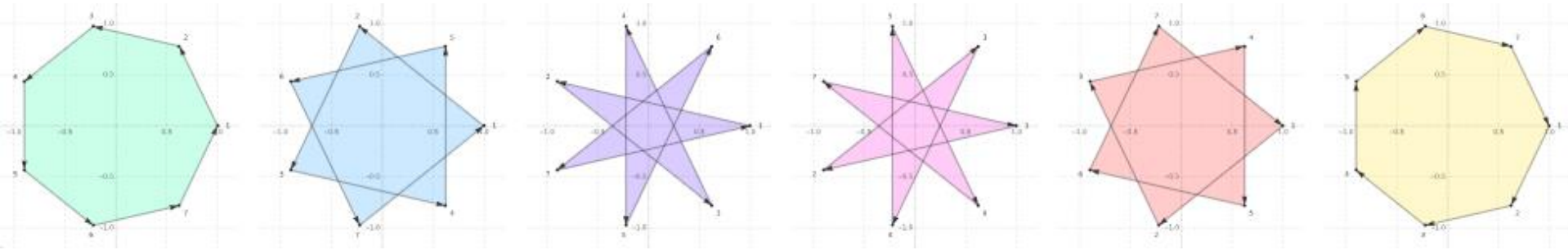



- 2D … dinamično programiranje
- 3D?
- odvisnosti modeliramo z grafom (DAG)
- iščemo optimalno zaprtje (clousure)
- prevedemo na min cut
  - cena = (vsi+) – (izpuščeni+) – (potrebni-)
  - source -> positive,  negative -> sink
- max flow (Ford-Fulkerson)



# Osnovni večkotniki

Zapiši oglišča večkotnika kot premik in vsoto skaliranih in rotiranih (oglišč) osnovnih večkotnikov.



- kompleksna števila (točke)
- osnovni večkotnik
  - $\omega = e^{2*\pi/n}$ , $\omega^i = e^{2*\pi*i/n}$
  - $E_{i,n} = \{(\omega^i)^j = \omega^{i*j}, \quad j = 0 \ldots n-1\}$

$$P = (x_t, y_t) + \sum_{i=1}^{n-1} s_i \operatorname{rot}(E_{i,n}, \varphi_i)$$

$$P = Wx = \begin{bmatrix} 1 \\ E_{1,n} \\ E_{2,n} \\ E_{3,n} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = [\omega^{ij}]x$$

- x … kompleksna števila
- $x_0$ = translacija (dx = realno, dy = imaginarno)
- $x_1$, …, $x_n$ = skaliranje (magnituda), rotacija (kot)

- Diskretna Fourierova transformacija (W … DFT matrika)
  - ω = n-ti koren enote
- inverzen FFT
  - Cooley-Tukey ($n = 2^k$)
  - Bluestein (poljuben n) … konvolucija dveh seznamov (zero padding)