

# Univerzitetni programerski maraton

**FINALE 2013 – rešitve nalog**

# Snežna odeja

Nariši 3 cm snega nad ASCII pokrajino.

- poišči najvišji '#' v vsakem stolpcu
- pazi na prazne stolpce

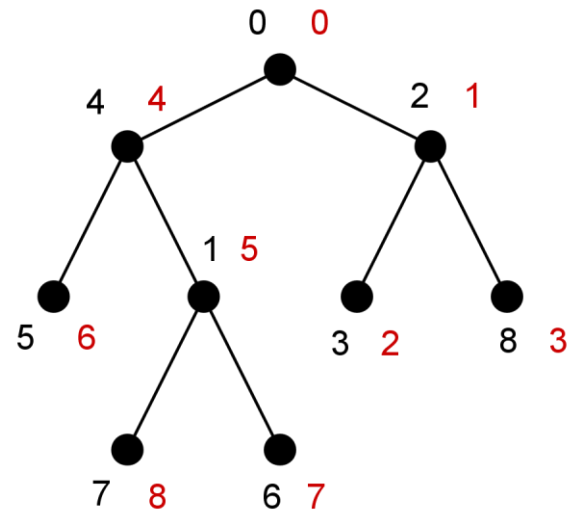
```
.....
.....
.....
.####.....
#####..#..
.####..#.#.
..#...#...#
.###...#####

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

# Jame

Vozlišča v drevesu preiskujemo na dva različna načina. S katerim bomo prej prišli do ciljnega vozlišča?

- oštevilčimo točke z DFS
  - rekurzija



- primerjamo podano in DFS številko točke

# Izjavne formule

---

Kakšna je verjetnost, da bo logični izraz resničen, če so vrednosti spremenljivk naključne?

- parsanje izraza  $(A \mid (C \& D)) \Leftrightarrow (B \mid E) \Leftrightarrow C \& D \Rightarrow E$ 
  - rekurzivno
  - zunanji operator z min. prioriteto
- testiranje vseh možnih vrednosti
  - rekurzivno
  - bitne maske

# Izjavne formule

---

```
class St:
    def __init__(self,x): self.x=x
    def __invert__(self): return St(not self.x)
    def __mul__(self,other): return St(self.x and other.x)
    def __add__(self,other): return St(self.x or other.x)
    def __rshift__(self,other): return St((not self.x) or other.x)
    def __eq__(self,other): return St(self.x == other.x)
    def __str__(self): return str(self.x)

izraz = raw_input().strip()
izraz = izraz.replace('&','*').replace('|','+')
izraz = izraz.replace('<=>','==').replace('=>','>>')
st = 0
for mask in range(2**10):
    val = izraz
    for c in reversed(range(10)):
        val = val.replace(chr(ord('A')+c),"St(%s) "%((mask&(1<<c))!=0))
    st += eval(val).x
print 1.0*st/2**10
```

# Periodična števila

Seštej dve neskončno dolgi periodični števili.

$$(45)3 + (455)51 = (009100)4$$
$$\begin{array}{r} \dots 54545453 \\ + \dots \underline{54554551} \\ \dots 00091004 \end{array}$$

- kako dolga je lahko perioda?  $O(p^2)$
- kaj pa neperiodičen del?  $O(n+p)$

# Periodična števila

---

- obrnemo nize

`s[i]` namesto `s[s.size()-i]`

- poravnamo neperiodične dele

(54)53

(455)51

- seštejemo periodične dele

$(x\%p_1, x\%p_2, c)$

$$\begin{array}{r} 4\ 5\ 4\ 5\ 4\ 5\ 4\ 5\ 4 \\ +\ 1_1\ 4_1\ 5_1\ 5_1\ 4_1\ 5_0\ 5_1\ 4_1\ 5_1\ 5 \\ =\ 9\ 1\ 0\ 0\ 0\ 0\ 9\ 1\ 0 \end{array}$$

# Periodična števila

---

- normaliziramo rešitev

- manjša perioda je vedno delitelj osnovne

$$a * p_1 + b * p_2 = \text{gcd}(p_1, p_2)$$

$$(123123123) \rightarrow (123)$$

- zamaknemo periodo

$$(000910)04 \rightarrow (009100)4$$

- alternativna/tekmovalna rešitev

- seštevamo do nekaj tisoč števk

- poiščemo periodo in odmik



# Periodična števila

---

```
import sys
pa,a = raw_input().strip()[1:].split(' ')
pb,b = raw_input().strip()[1:].split(' ')
while len(a)<3000: a=pa+a
while len(b)<3000: b=pb+b
c = str(int(a)+int(b))
while len(c)<3000: c='0'+c

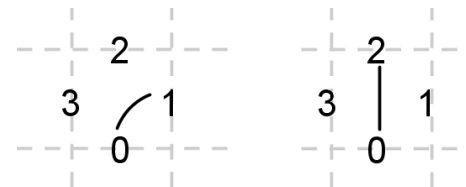
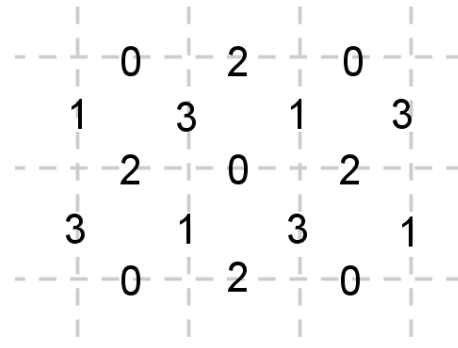
for p in xrange(1,1000):
    for i in xrange(1,1000):
        ok = True
        for j in xrange(i,3000-p+1):
            if c[-j]!=c[-(j+p)]:
                ok = False
                break
        if ok:
            per = c[len(c)-i-p+1:len(c)-i+1]
            suf = c[len(c)-i+1:]
            print '(%s)%s'%(per,suf)
            sys.exit(0)
```

# Puzzle

Iz danega števila ravnih ( $s$ ) in ovinkastih ploščic ( $t$ ) sestavi krožno pot.

- $t$  je sod
  - na začetku in na koncu moramo gledati v isto smer

- $s$  je sod



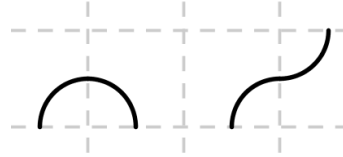
$$d - l = \pm 4$$

$$d \cdot (1) + l \cdot (-1) + s \cdot 2 = 0 \pmod{4}$$

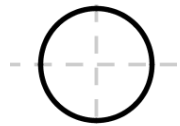
# Puzzle

- $s = 0$

$$t = k \cdot 4$$



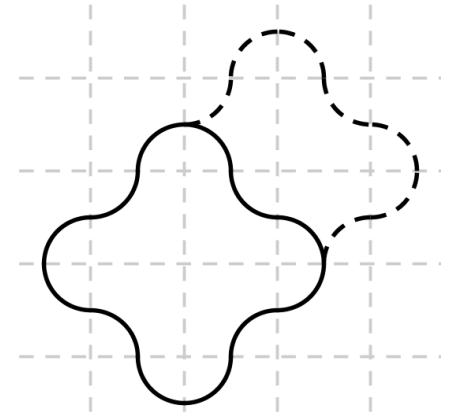
$$t = 4$$



$$t = 8$$

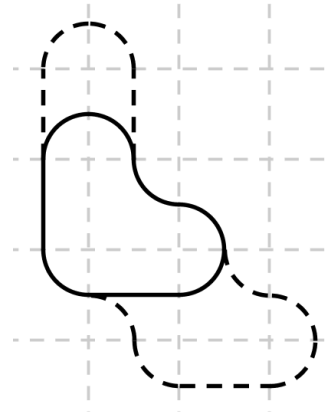
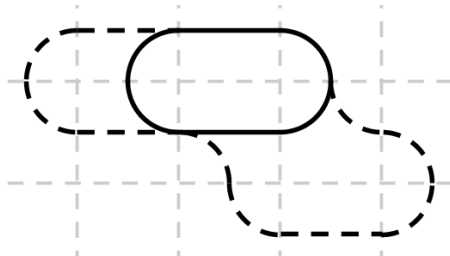
ne obstaja

$$t = 12 + k \cdot 4$$



- $s \neq 0$

$$t = 4 + k \cdot 4$$



$$t = 6 + k \cdot 4$$

# Puzzle

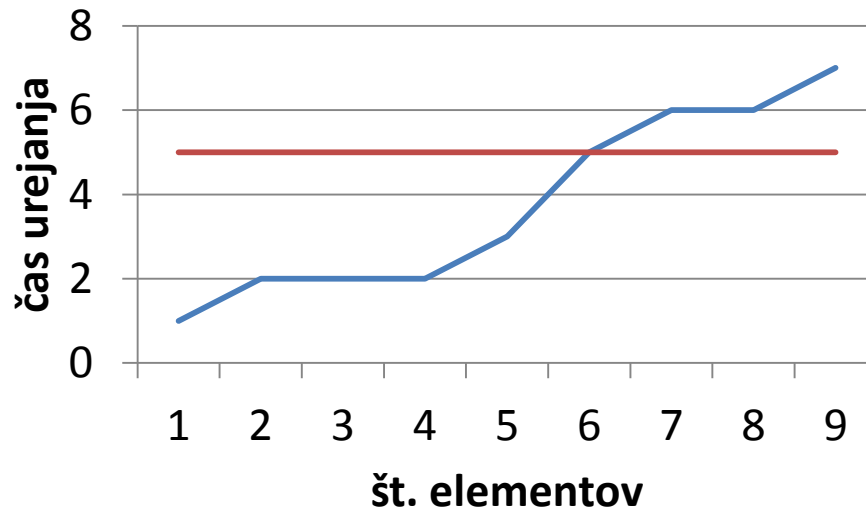
---

```
s,t = map(int,raw_input().strip().split())
sol = ""
if s==0:
    if t==4: sol="RRRR"
    else:
        ext = "LR"*((t-12)/4)
        sol = ext + "RLR" + ext[::-1] + "RLRRLRRLR"
else:
    if t%4==0:
        ext_s = "F"*(s/2)
        ext_t = "LR"*((t-4)/4)
        sol = ext_s+"R"+ext_t+"RF"+ext_t[::-1]+ext_s[:-1]+"RR"
    else:
        ext_s = "F"*((s-2)/2)
        ext_t = "LR"*((t-6)/4)
        sol = "F"+ext_s+"RR"+ext_s+"LR"+ext_t+"RF"+ext_t[::-1]+"R"
print sol
```

# Vzporedno urejanje

Optimalno razdeli urejanje števil med več različno hitrih računalnikov.

- koliko časa potrebujemo za urediti  $n$  elementov?
- koliko elementov lahko uredimo v času  $t$ ?
  - bisekcija po času



# Vzporedno urejanje

---

- računalniki so neodvisni
- koliko elementov uredi rač. s hitrostjo  $f$  v času  $t$ ?
  - zopet bisekcija po št. elementov  $k$

$$\frac{k \cdot \lceil \log_2 k \rceil}{f} \leq t$$

- s kakšnimi števili imamo opravka?
  - max  $t$   $n \log_2 n > 2^{31}$
  - max št. elementov  $k$   $r n > 2^{31}$

# Vzporedno urejanje

---

- pravilno, narobe, skoraj pravilno?

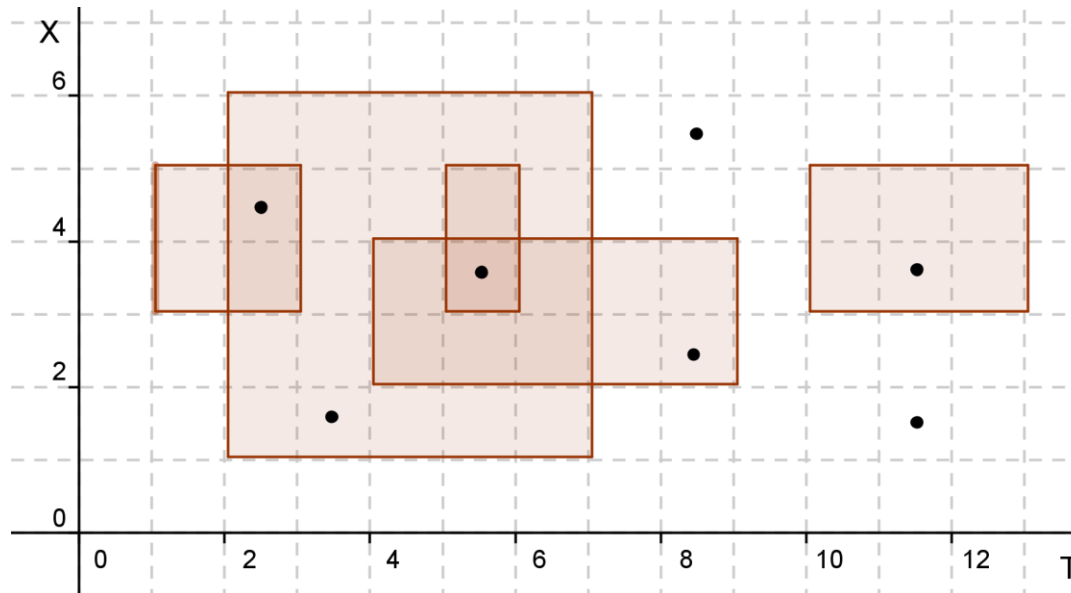
```
double mt=0, Mt=1e9*32;
while (Mt-mt>1e-9) {
    double t=(mt+Mt)/2;
    int k=koliko(t);
    if (k>=n) Mt=t;
    else mt=t;
}
print ("%f",Mt);
```

```
double mt=0, Mt=1e9*32;
for (int i=1;i<=100;i++) {
    double t=(mt+Mt)/2;
    long long k=koliko(t);
    if (k>=n) Mt=t;
    else mt=t;
}
print ("%f",Mt);
```

# Varnost pri delu

Na strnjenih odsekih AC izvajajo delavci ob različnih urah popravila. Koliko jih dela ob danem času na danem odseku?

- 2D problem – čas, lokacija



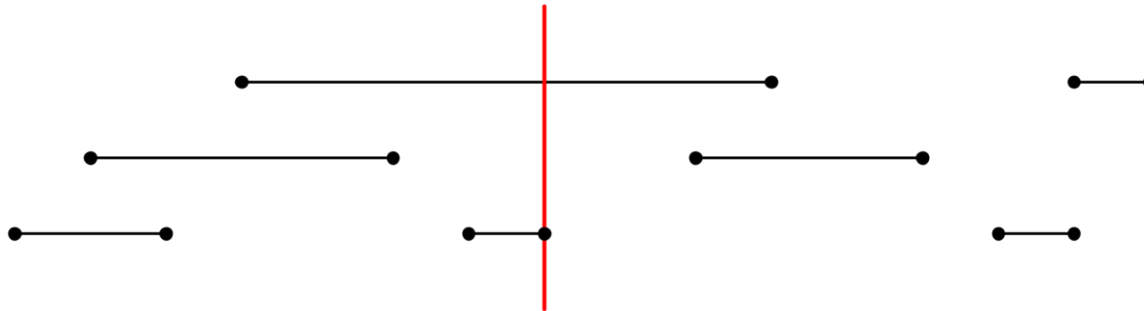
naloga Kolaž (2. kolo 2013)?



# Varnost pri delu

---

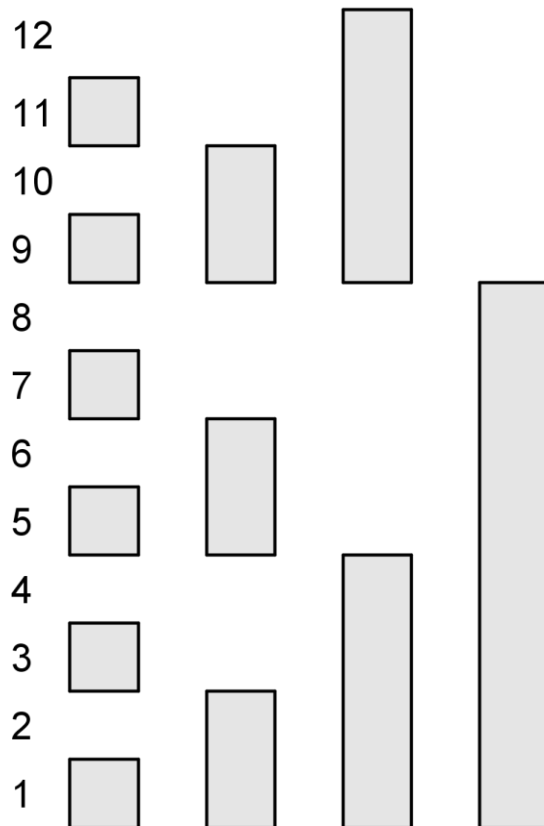
- dinamično dodajanje intervalov s poizvedbami
  - sortiramo dogodke po času
  - simuliramo dogajanje



- drevo začetkov, koncev
  - AVL, Red-Black, ...

# Varnost pri delu

- kompresija koordinat
- Fenwick tree



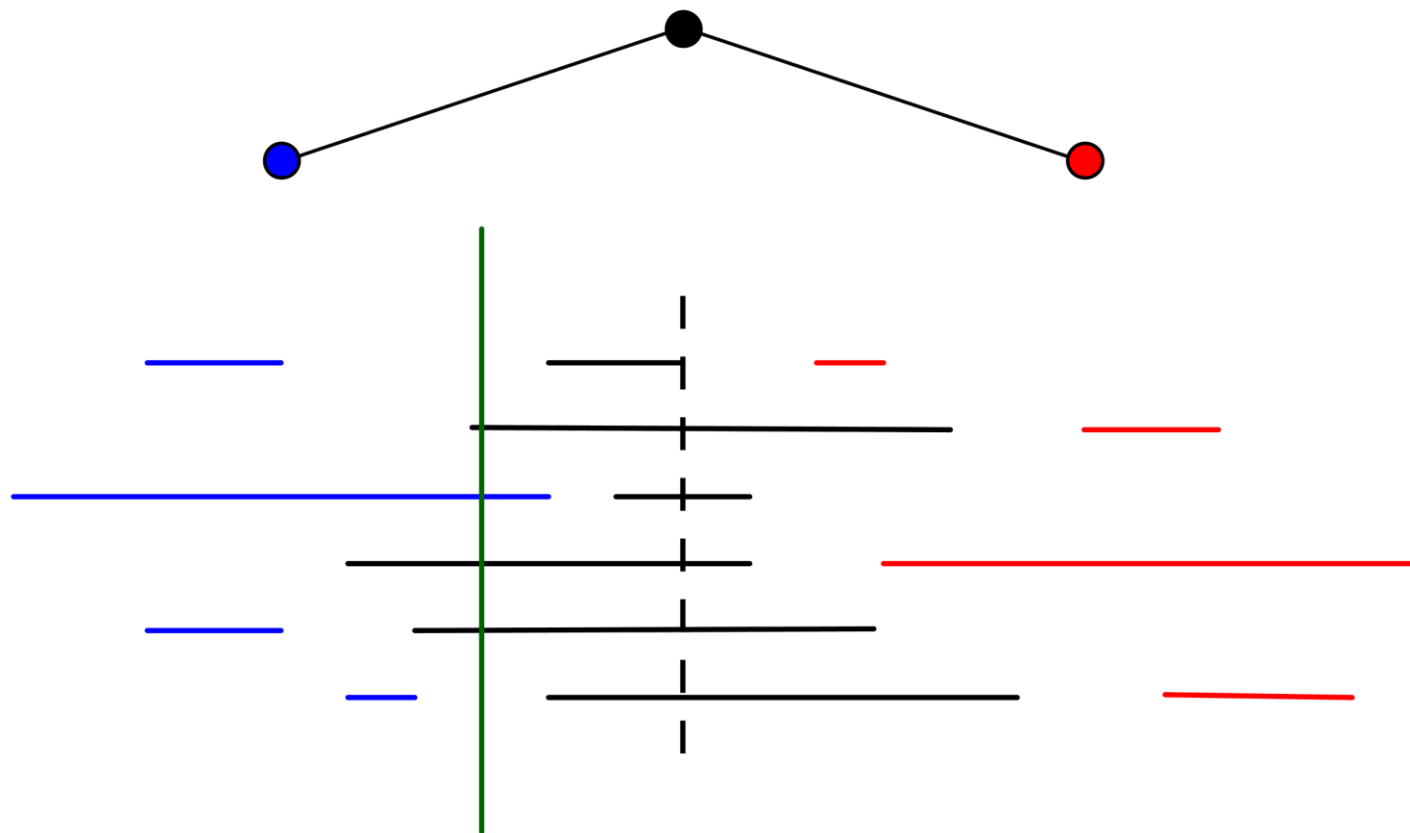
```
int tree[N+1];

void update(int v, int d) {
    for (int x=v; x<=N; x+=x&-x)
        tree[x]+=d;
}

int query(int v) {
    int q=0;
    for (int x=v; x>0; x-=x&-x)
        q+=tree[x];
    return q;
}
```

# Varnost pri delu

- Interval tree

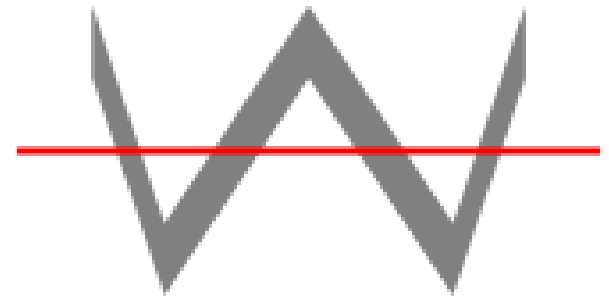


# Arbitraža

---

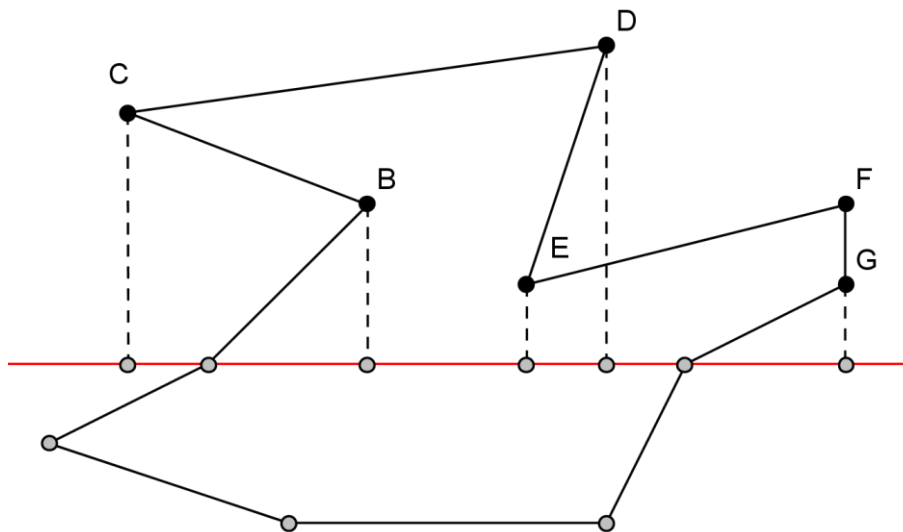
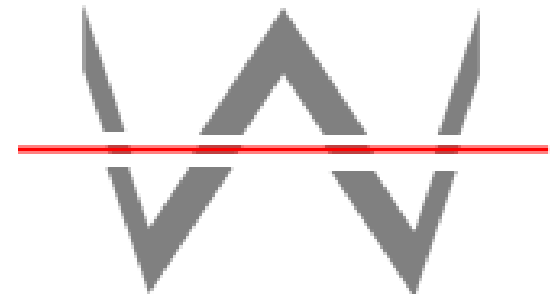
Večkotnik s premico razdeli na dve območji in izračunaj njuni površini.

- enačba premice
  - $(x_1, y_1), (x_2, y_2) \rightarrow ax + by + c = 0$
- presečišče daljice in premice
  - 2 enačbi, 2 neznanki
  - kolinearnost
  - ali je rezultat na daljici?



# Arbitraža

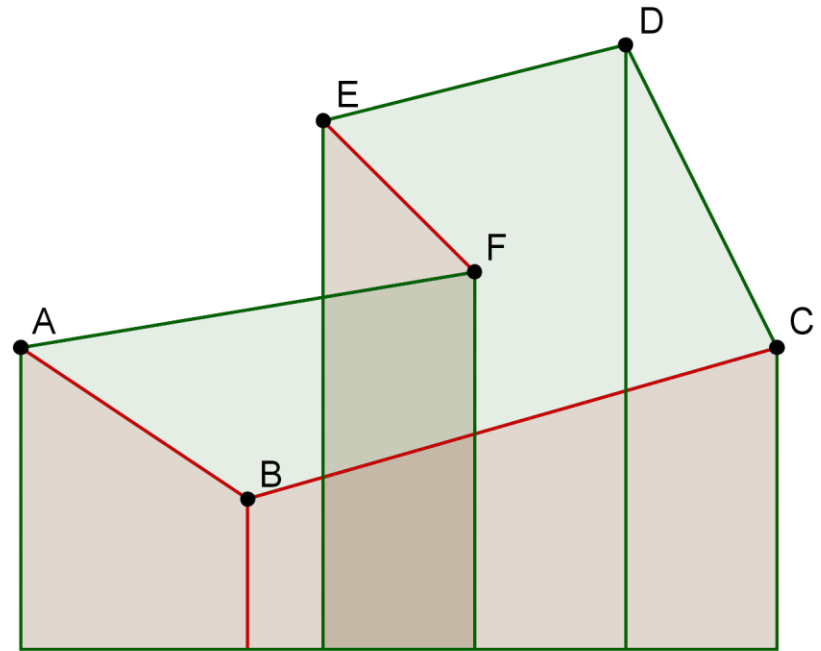
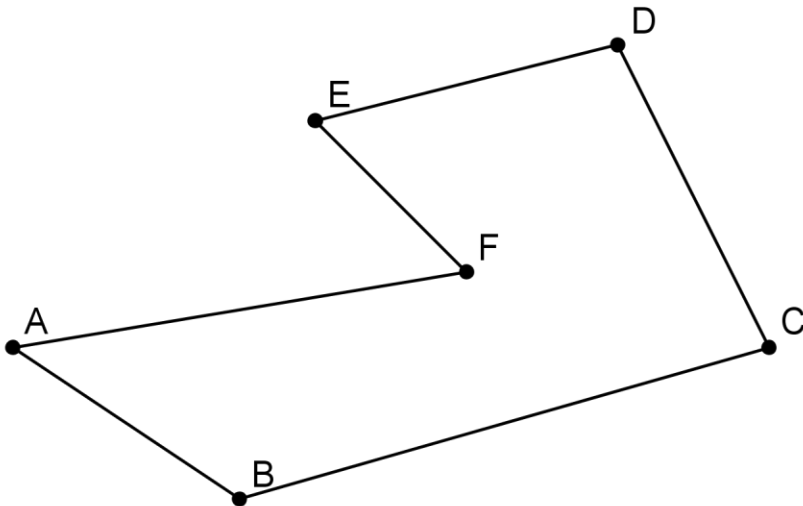
- kako poiščemo in “zapremo” posamezne kose?
  - težko
- projekcije točk na premico
  - ni težav s povezanostjo
  - skalarni produkt



# Arbitraža

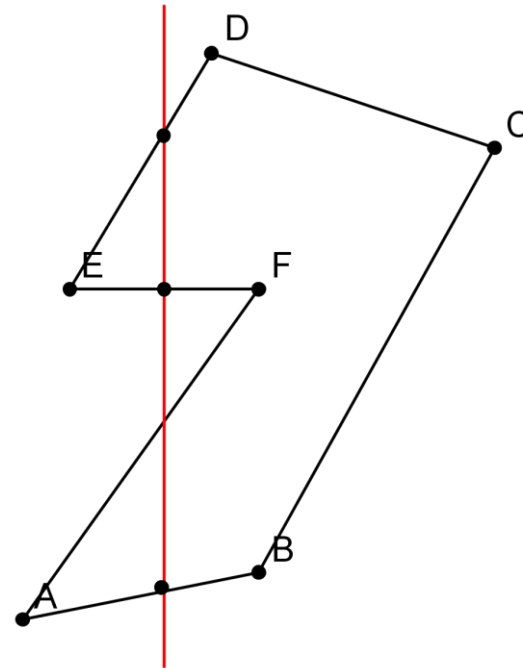
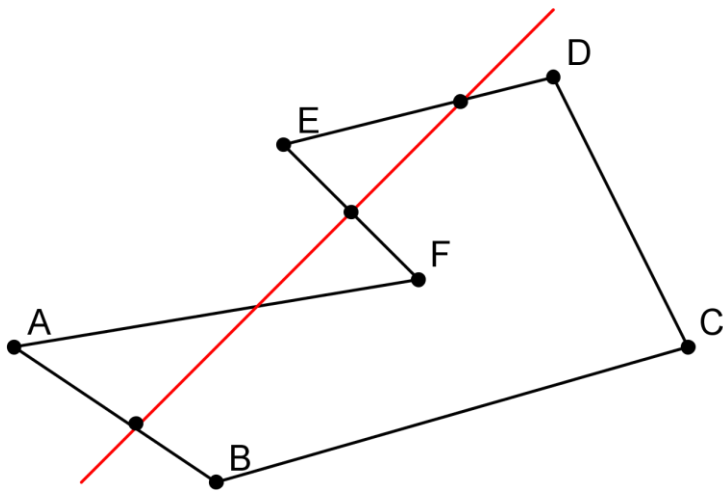
- ploščina večkotnika

$$A = \left| \frac{1}{2} \sum_{i=0}^{n-1} (x_i y_{i+1} - x_{i+1} y_i) \right|$$



# Arbitraža

- alternativna rešitev:
  - “stran” točk
  - rotacija večkotnika



# Eurobasket

Iz seznama prihajajočih tekem izračunaj, katere ekipe še lahko dosežejo 1. mesto.

- Baseball Elimination Problem

ekipa	točke	tekme	ATL	PHI	NY	MON
Atlanta	83	8	0	1	6	1
Philadelphia	80	3	1	0	0	2
New York	78	6	6	0	0	0
Montreal	77	3	1	2	0	0



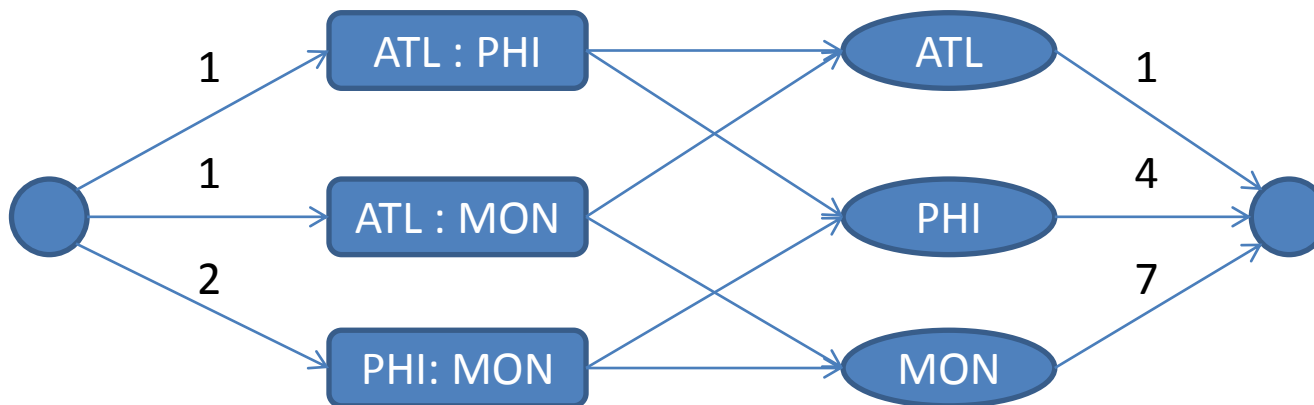
# Eurobasket

---

- rešujemo za vsako ekipo posebej
  - zmaga vse svoje tekme
  - preostale tekme razporedimo tako, da nobena ekipa ne preseže naše
- kako razporediti tekme?
  - heuristike
  - maksimalen pretok

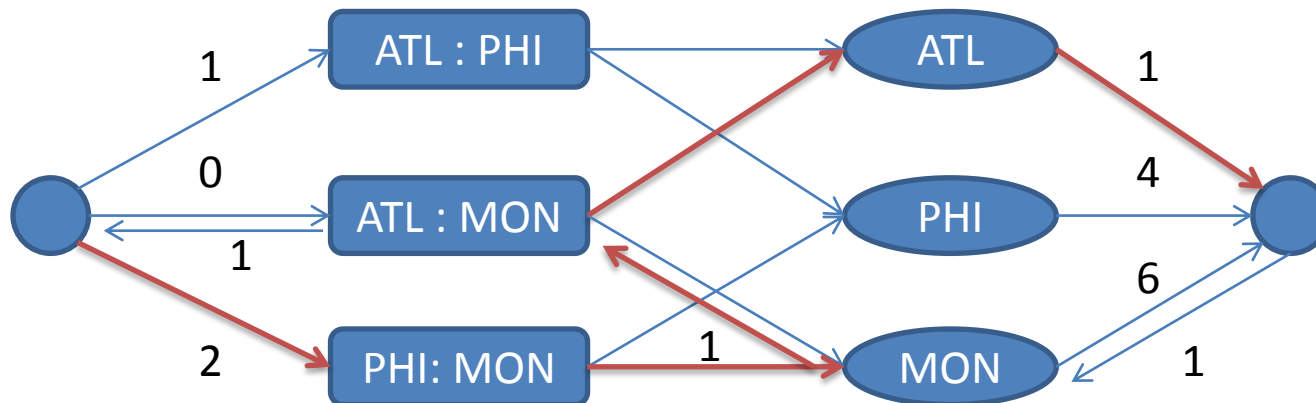
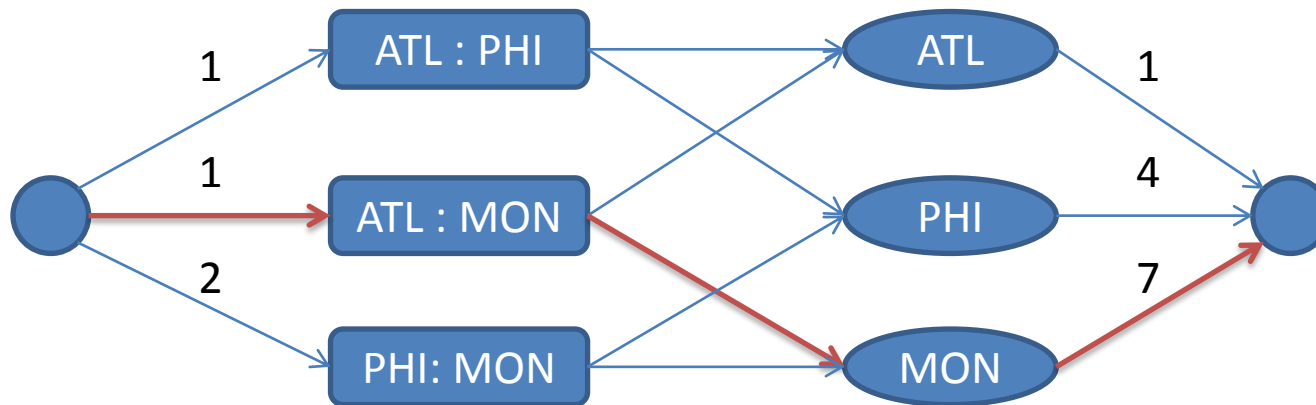
# Eurobasket

ekipa	točke	tekme	ATL	PHI	NY	MON
Atlanta	83	8	0	1	6	1
Philadelphia	80	3	1	0	0	2
New York	78	6	6	0	0	0
Montreal	77	3	1	2	0	0



# Eurobasket

- Ford-Fulkerson



# Eurobasket

---

- časovna zahtevnost

$$O(n \cdot f \cdot e) \approx 50 \cdot 1000 \cdot (50 \cdot 50 \cdot 3) \approx 4 \cdot 10^8$$

- implementacija

- seznam povezav
- obratne povezave
- sosednje povezave

```
int flow(int x, int f=inf) {
    if (mark[x]==m) return 0;
    mark[x]=m;
    if (x==sink()) return f;
    for (int i=0;i<adj[x].size();i++) {
        int e=adj[x][i], y=edge[e];
        if (cap[e]<=0) continue;
        int ff=flow(y,min(f,cap[e]));
        if (ff>0) {
            cap[e]-=ff;
            cap[rev(e)]+=ff;
            return ff;
        }
    }
    return 0;
}
```

# Zaključek

---

- knjige, izpiski (team notebook)
- ekipno reševanje
- testiranje, debugiranje
  
- trening
  - arhivi
  - tekmovanja
- krožek/priprave?