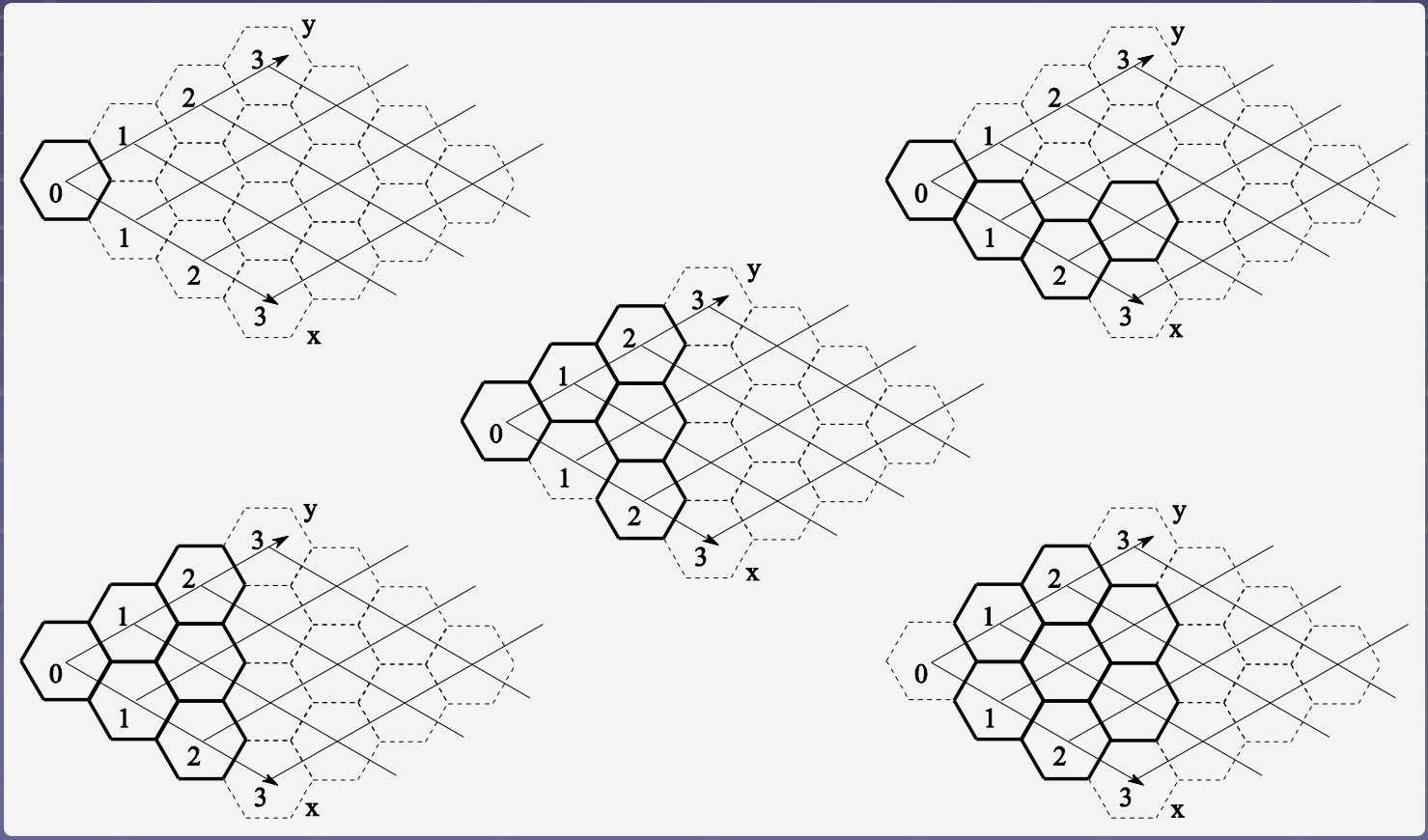
Sample Solutions

CTU Open Contest 2014

Czech Technical University in Prague

KOLONIE



Kolonie – základní úvaha

- Komplexy lze spojovat libovolně…
- … a ve vesmíru je místa dost.

- => Jde spojit cokoli s čímkoli pouze jednou stranou
  (na tvaru vlastně moc nezáleží)

Kolonie – postup

Z toho plyne:

- Jedna buňka: 6 oken
- Dvě buňky: 10 oken
- n buněk: 6.n – 2 * počet spojů

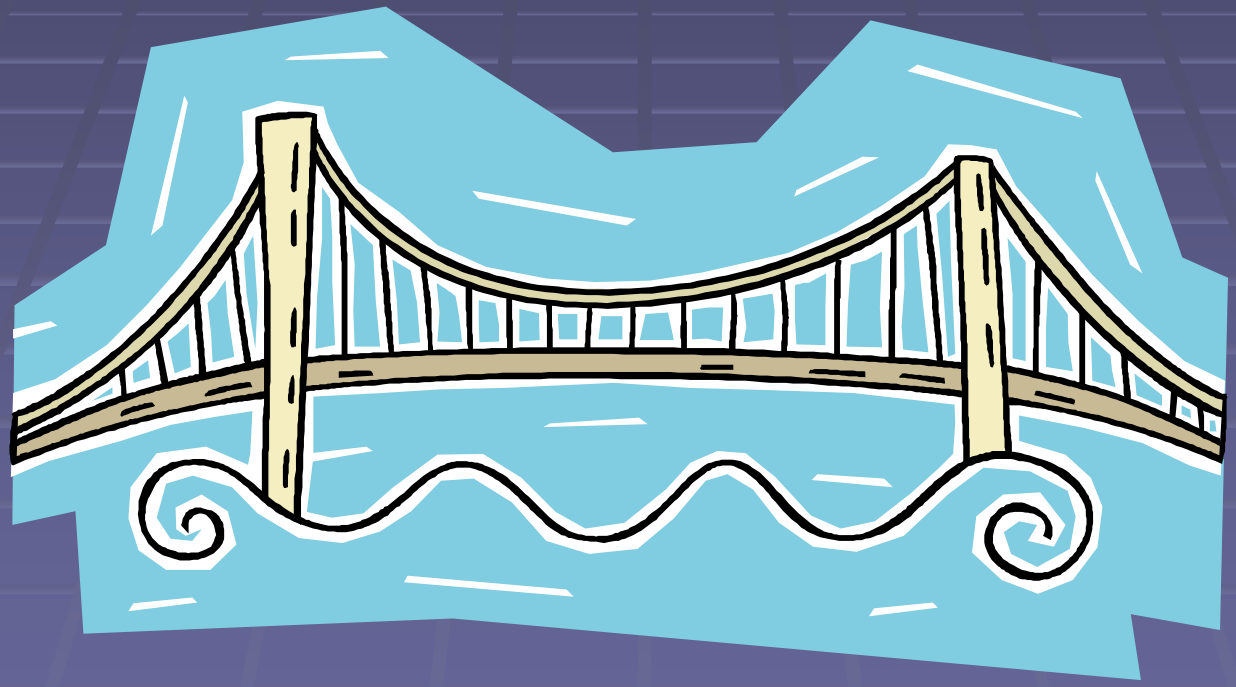
- => Při čtení buněk počítáme sousedy, za každého odečteme 2 okna

Kolonie – princip řešení

- Jakékoli 2 komplexy s **a** a **b** okny můžeme spojit a získat **a+b-2** oken

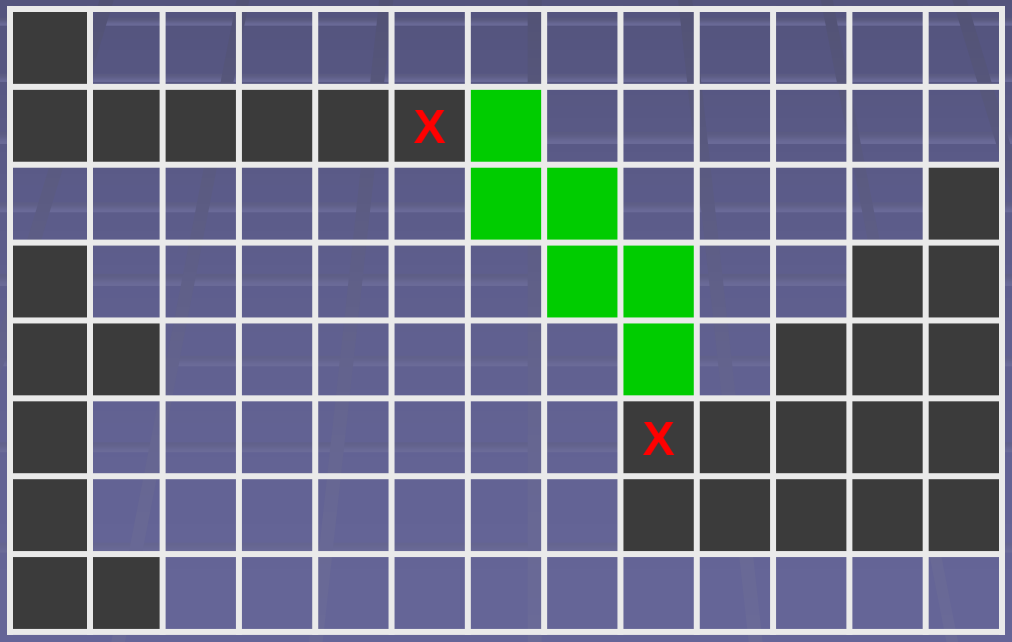
- … pak už to „jen“ implementovat …

MOST



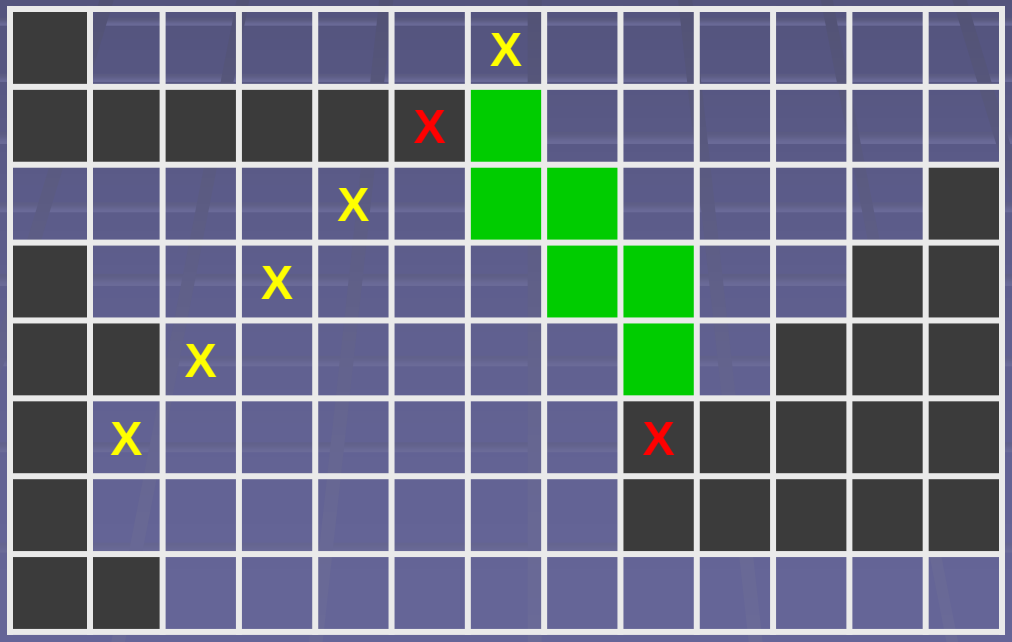
Most – myšlenka

- Jak obecně vypadá optimální spoj?



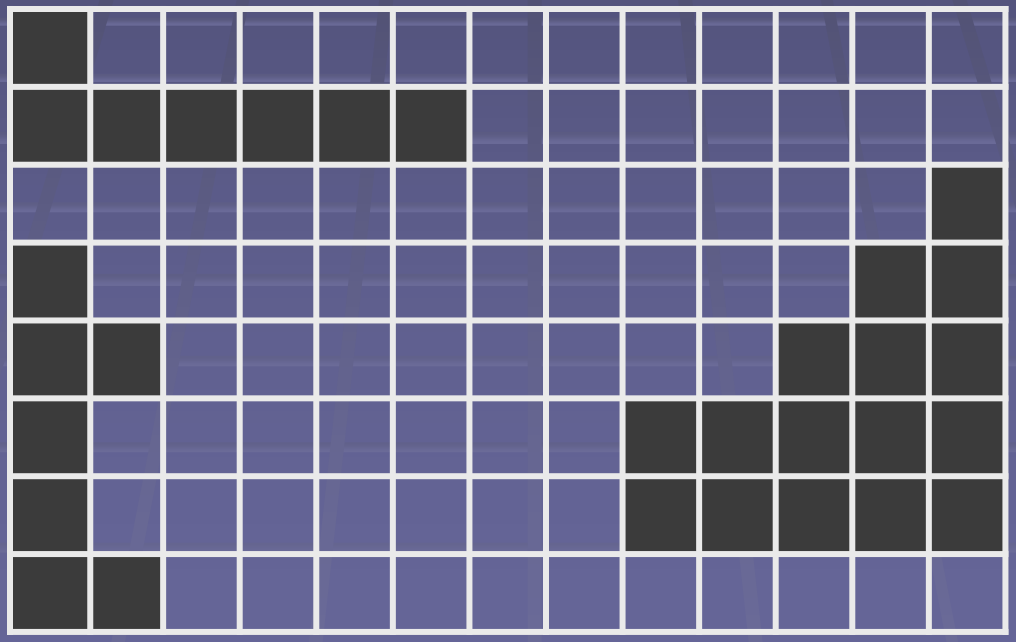
Most – myšlenka

- Každý řádek způsobí o 1 delší spoj
  - => „diagonála“ je stejně daleko



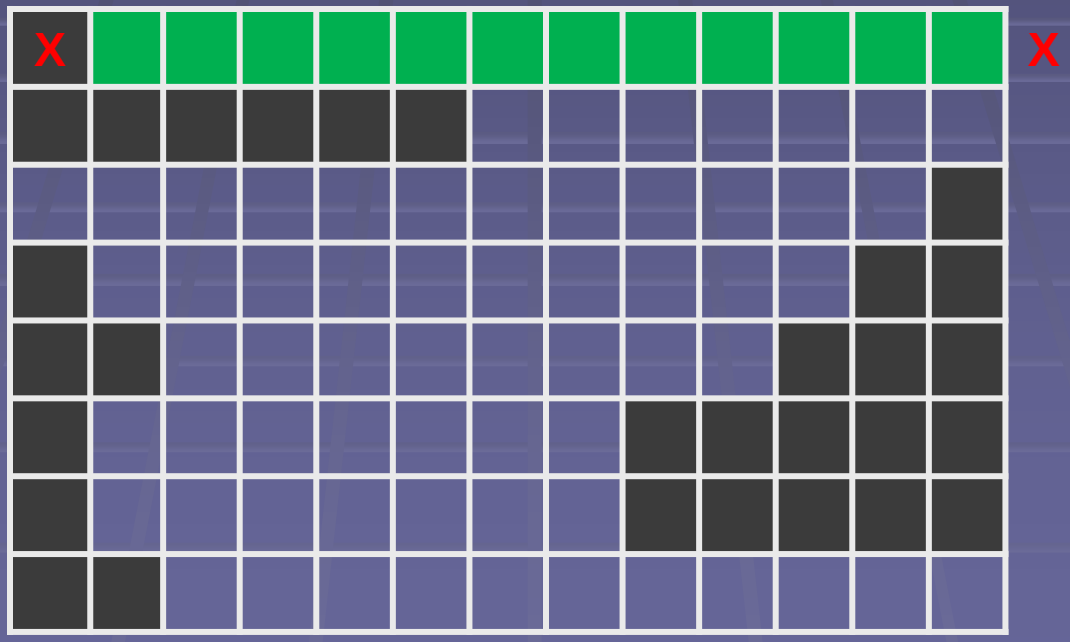
Most – princip

- => Stačí si pamatovat nejlepší možnost pro levý i pravý břeh



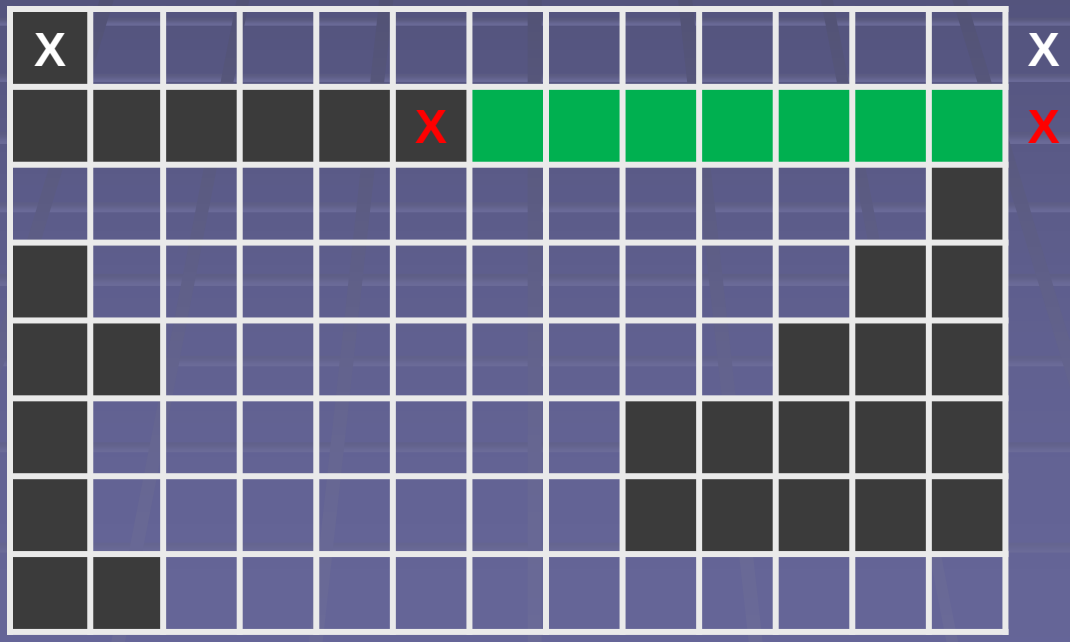
Most – princip

- Nejlepší pro levý i pravý břeh
  - S každým řádkem se o 1 „zhorší“



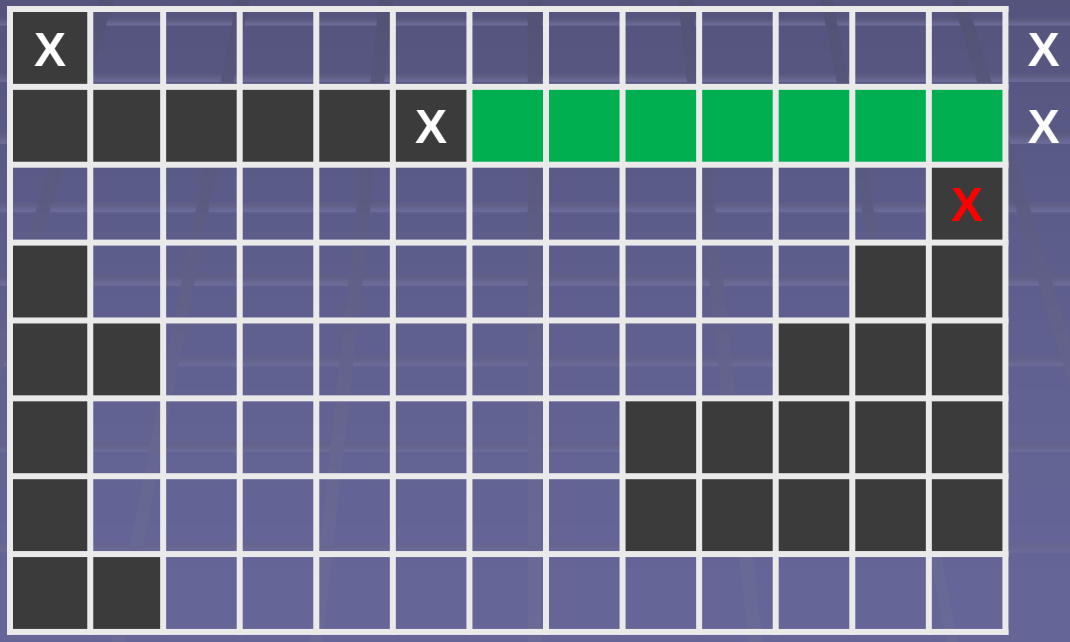
Most – princip

- Nejlepší pro levý i pravý břeh
  - S každým řádkem se o 1 „zhorší“



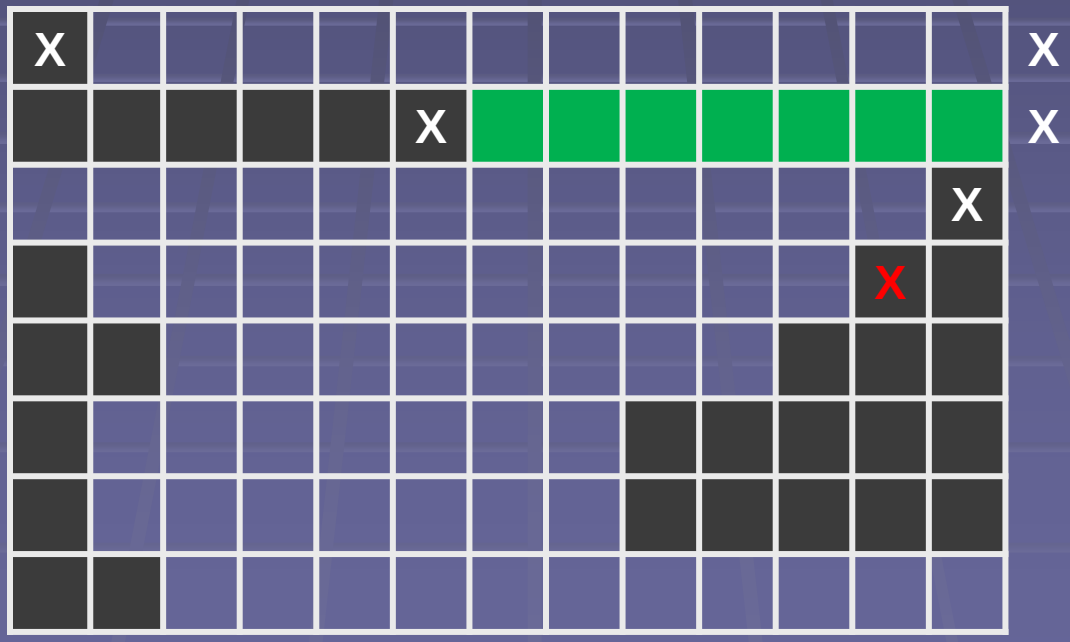
Most – princip

- Nejlepší pro levý i pravý břeh
  - S každým řádkem se o 1 „zhorší“



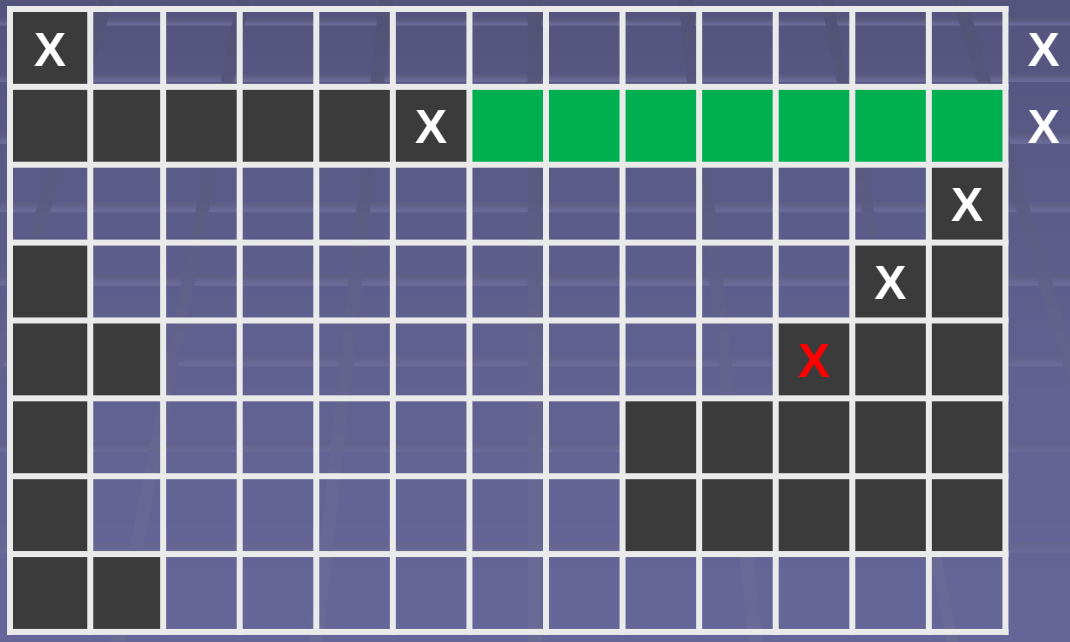
Most – princip

- Nejlepší pro levý i pravý břeh
  - S každým řádkem se o 1 „zhorší“



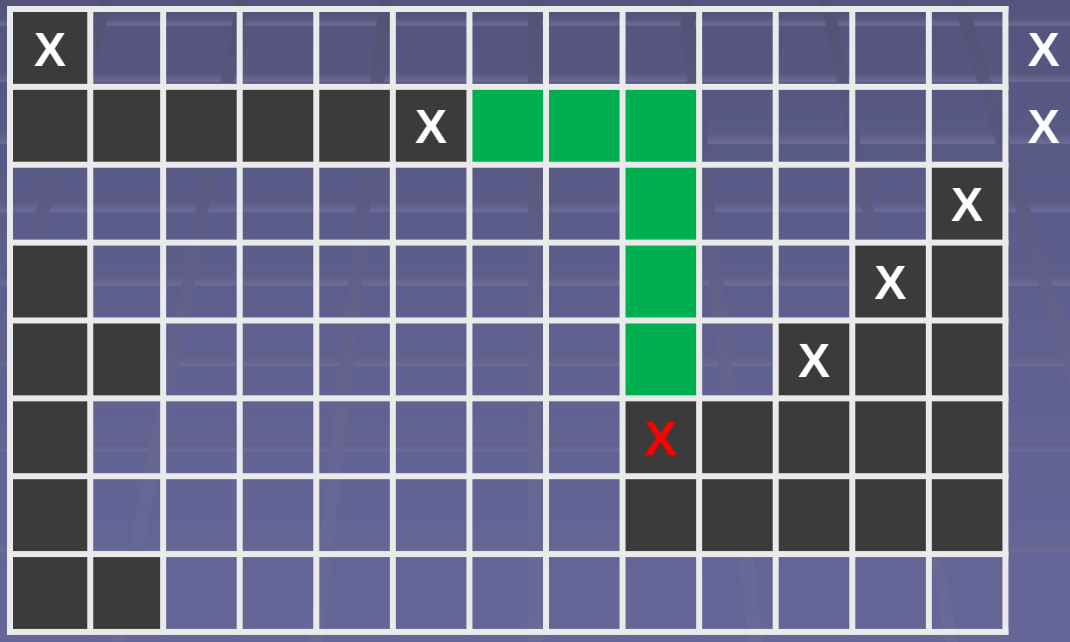
Most – princip

- Nejlepší pro levý i pravý břeh
  - S každým řádkem se o 1 „zhorší“



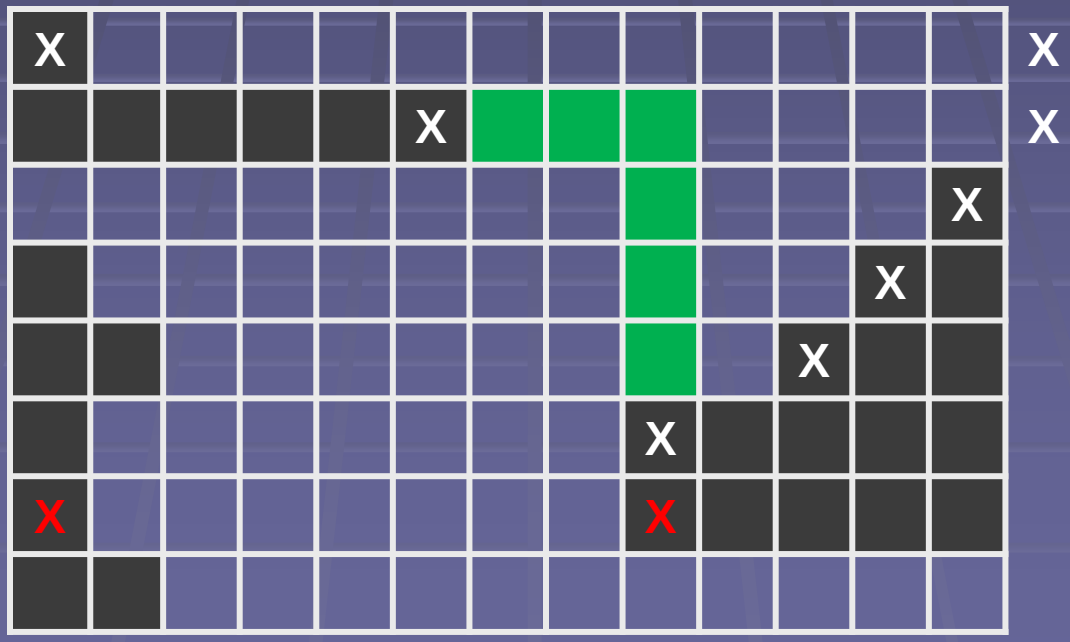
Most – princip

- Nejlepší pro levý i pravý břeh
  - S každým řádkem se o 1 „zhorší“



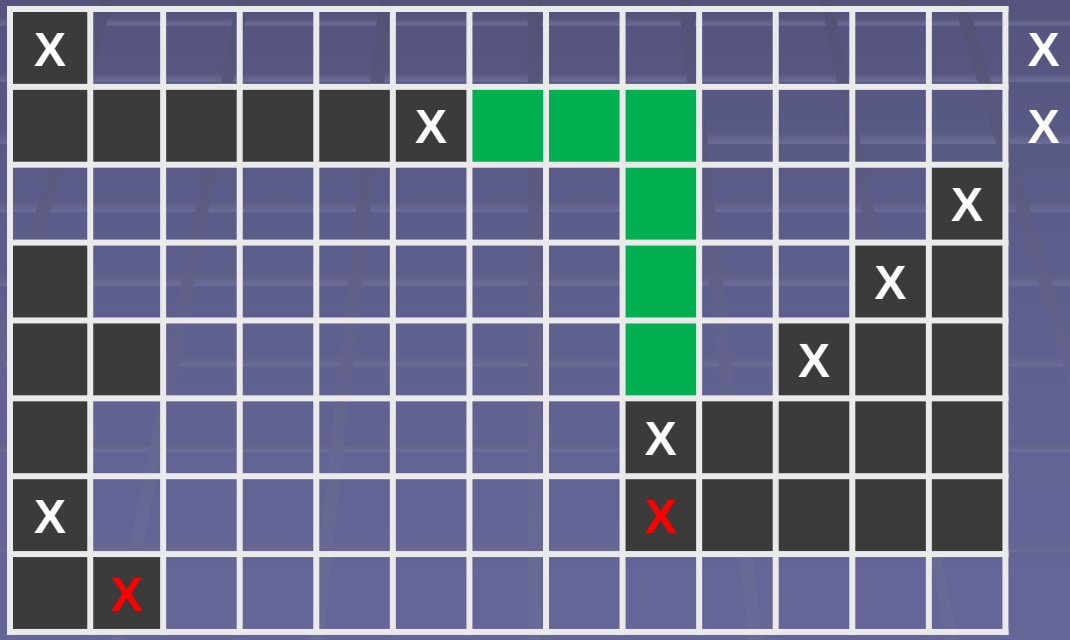
Most – princip

- Nejlepší pro levý i pravý břeh
  - S každým řádkem se o 1 „zhorší“



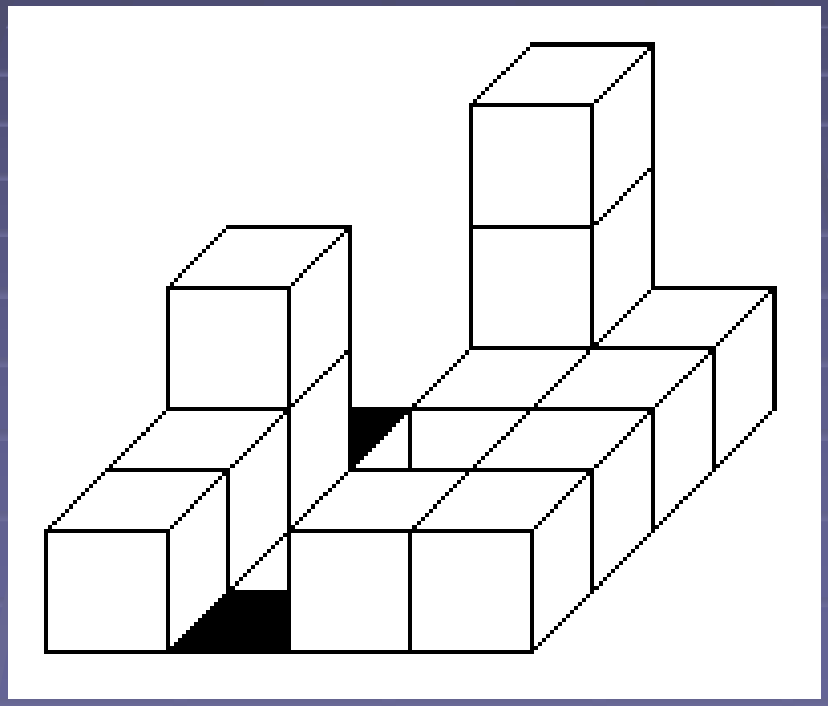
Most – princip

- Nejlepší pro levý i pravý břeh
    - S každým řádkem se o 1 „zhorší“



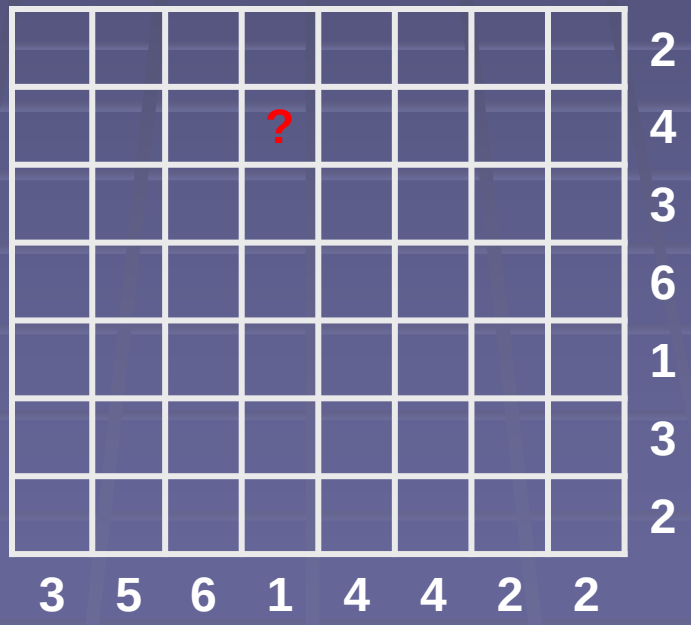
X													X
⬛	⬛	⬛	⬛	⬛	X	🟩	🟩	🟩					X
								🟩				X	
⬛								🟩			X	⬛	
⬛	⬛							🟩		X	⬛	⬛	
⬛								X	⬛	⬛	⬛	⬛	
X								X	⬛	⬛	⬛	⬛	
⬛	X												

STAVITEL



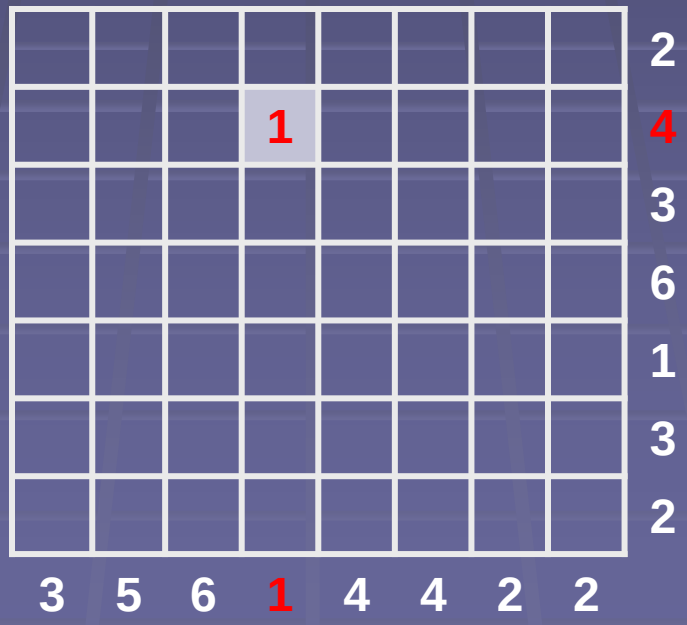
Stavitel – zadání

- Dosadit čísla, když známe maximum každého řádku a sloupce



Stavitel – jak na to?

- Maximum je docela jednoduché...
  - => dáme **minimum** obou čísel



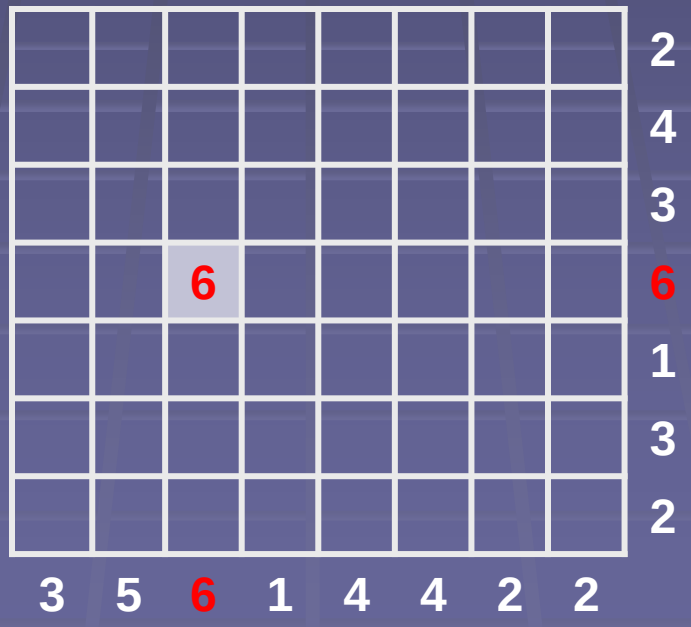
Stavitel – jak na to?

- Maximum je docela jednoduché...
  - => dáme **minimum** obou čísel

2	2	2	1	2	2	2	2	2
3	4	4	1	4	4	2	2	4
3	3	3	1	3	3	2	2	3
3	5	6	1	4	4	2	2	6
1	1	1	1	1	1	1	1	1
3	3	3	1	3	3	2	2	3
2	2	2	1	2	2	2	2	2
3	5	6	1	4	4	2	2	

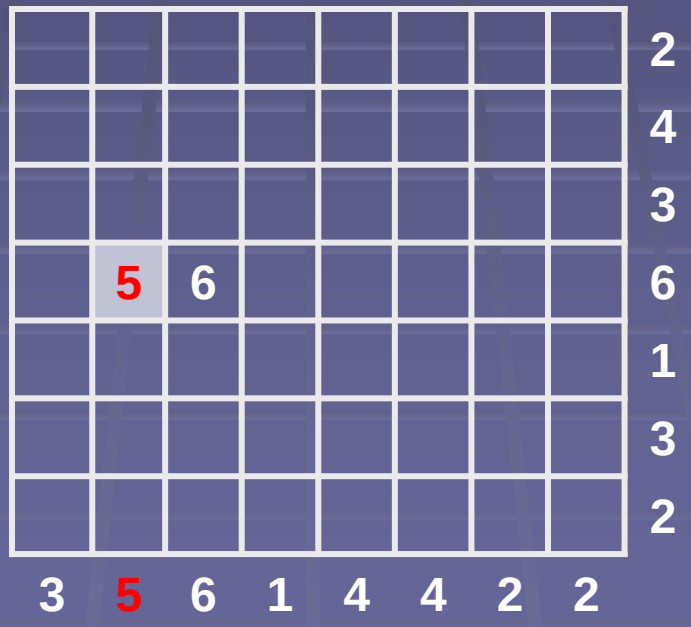
Stavitel – minimum

- Minimální budova je zajímavější...
  - Začneme od největšího čísla



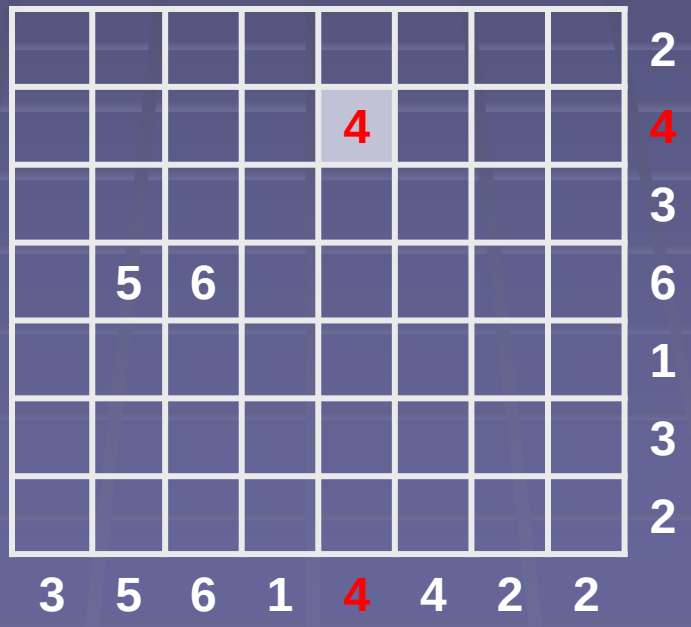
Stavitel – minimum

- Minimální budova je zajímavější...
  - Potom druhé největší…



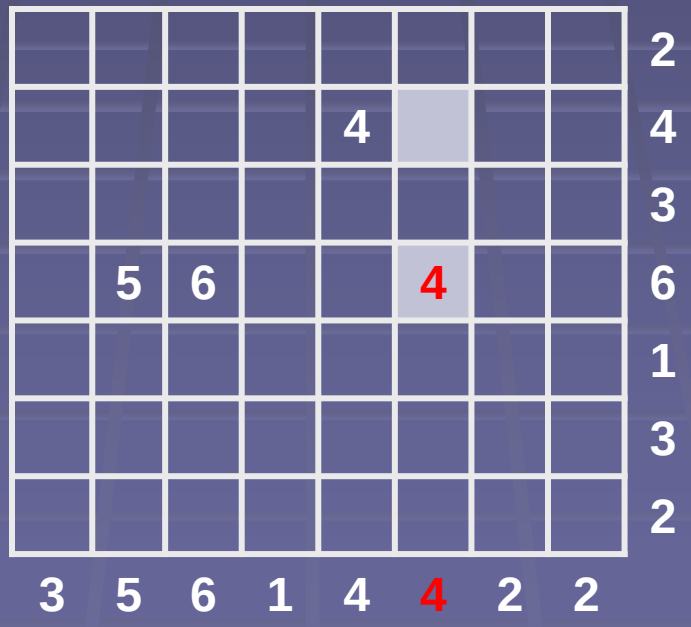
Stavitel – minimum

- Minimální budova je zajímavější…
  - A třetí…



Stavitel – minimum

- Minimální budova je zajímavější...
  - … to tam ale máme dvakrát …



Stavitel – dotažení

- … a tak dále …
- Ale ještě něco:
  Jak rychle hledat „největší číslo“?
- Na pořadí na vstupu ve skutečnosti vůbec nezáleží… ☺
  - (řádky/sloupce jsou nezávislé)

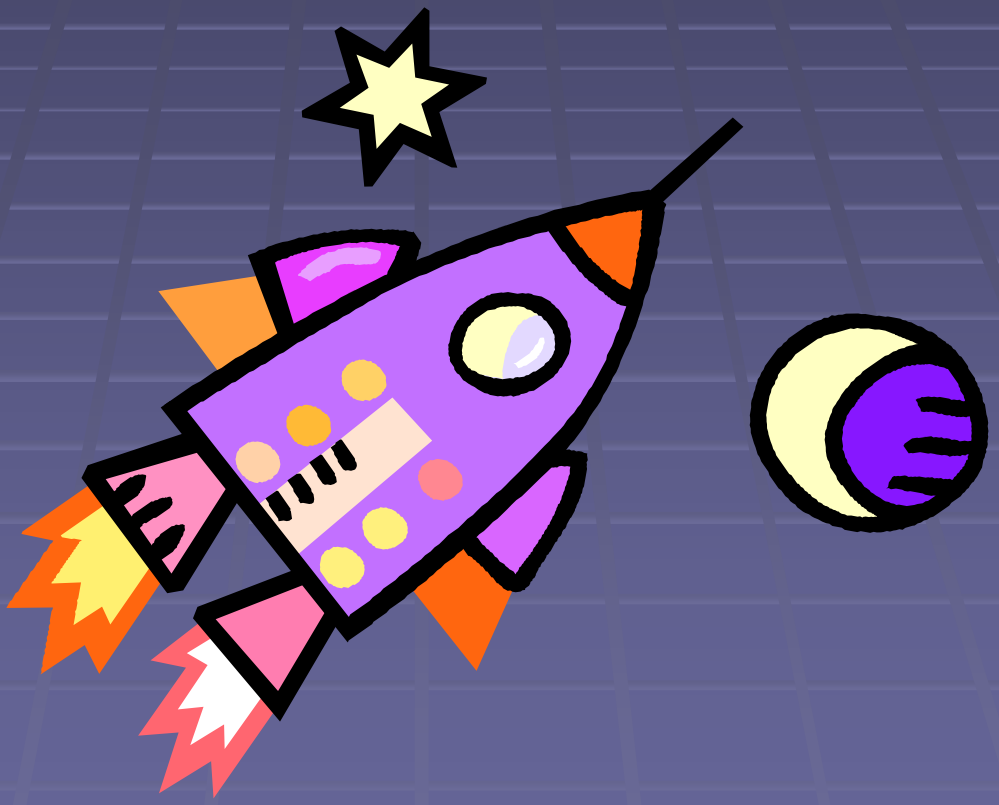
Stavitel – řešení

- Seřadíme oba seznamy dle velikosti
  - **6 5 4 4 3 2 2 1**
  - **6 4 3 3 2 2 1**
- Stejné první číslo:
  - Započítáme jednou a odebereme
- Různá první čísla:
  - Započítáme větší a odebereme

Stavitel – řešení

- Seřadíme oba seznamy dle velikosti
  - 6 5 4 4 3 2 2 1
  - 6 4 3 3 2 2 1
- Výsledek:
  - 6 + 5 + 4 +4 + 3 + 3 + 2 +2 + 1

LODĚ



Lodě – „myšlenka“

- Vyšší dimenze mají výrazně lepší poměr cena/hmotnost
  - Navíc jsou hmotnosti dělitelné:
    1 3 9 27 81 …

- => Místo 3 předmětů dimenze **n**
  bereme 1 předmět dimenze **n+1**

Lodě – řešení

- => Místo 3 předmětů dimenze **n**
  bereme 1 předmět dimenze **n+1**
- => Od každé dimenze maximálně **2ks**

- **Převod do trojkové soustavy… ☺**

VISIBLE

Visible – co s tím?

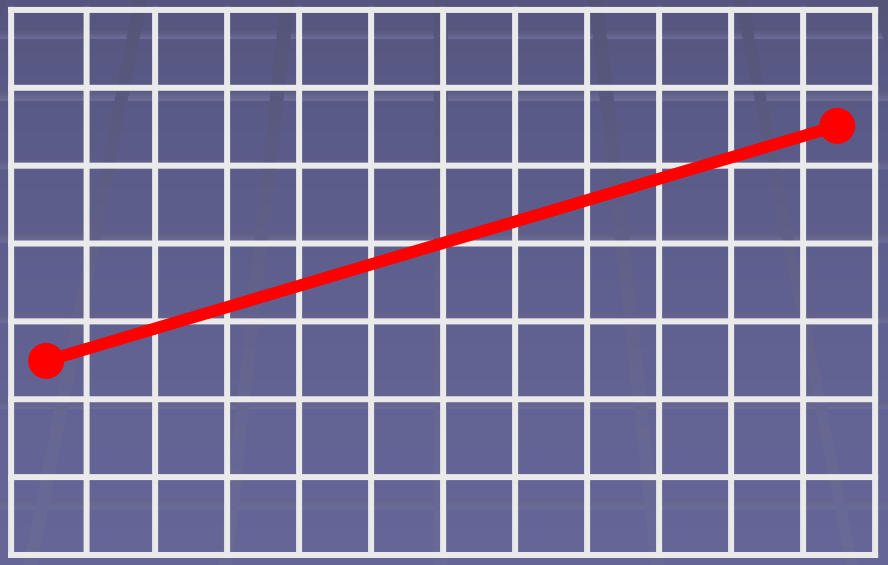
- „3D geometria je fujky fujky…“

  (Mimino, říjen 2014)

- … naštěstí je „jenom“ kostičková

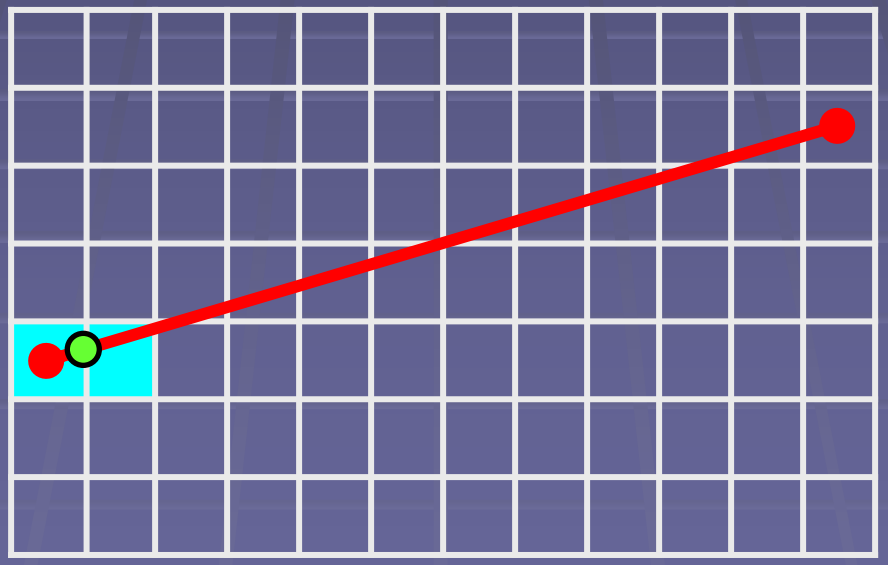
Visible – postup

- Které kostičky protíná úsečka
  - Stačí celočíselná aritmetika



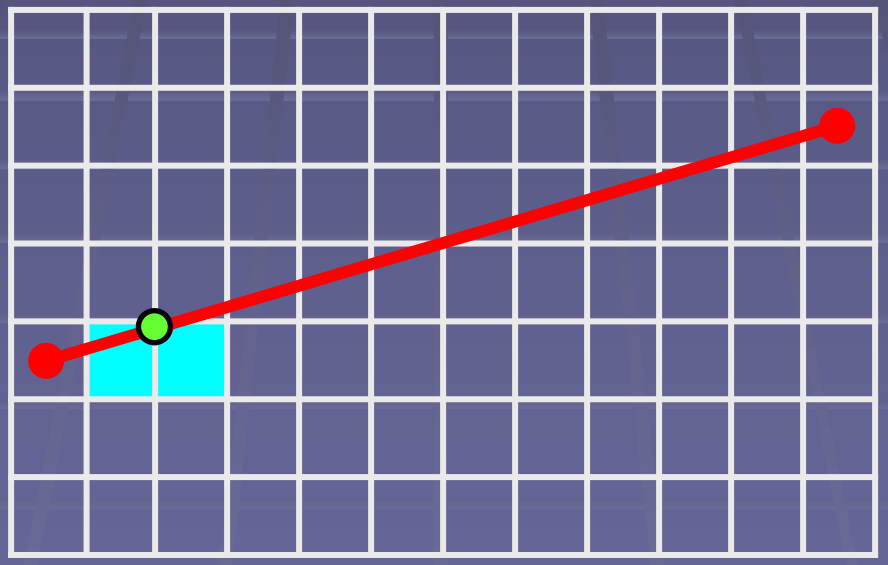
Visible – postup

- Které kostičky protíná úsečka
  - Porovnáme s výškou terénu



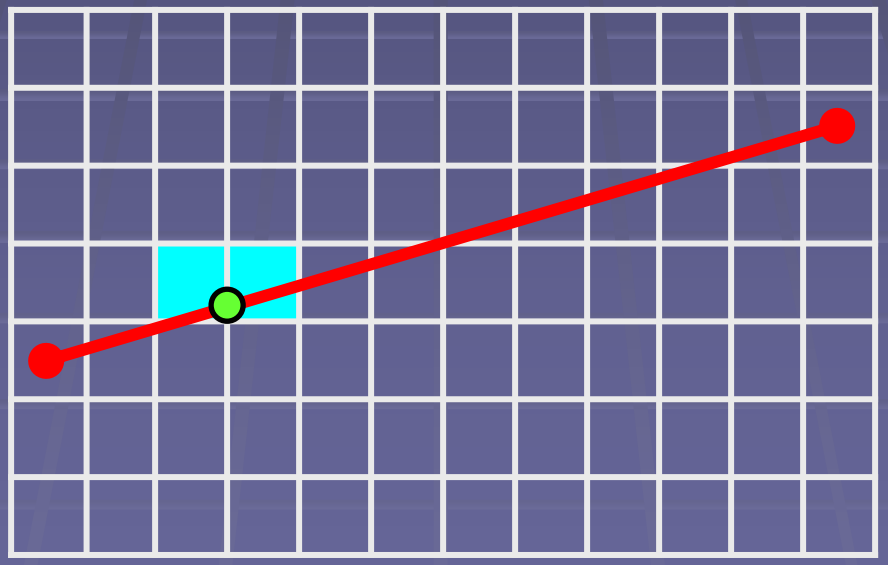
Visible – postup

- Které kostičky protíná úsečka
  - Porovnáme s výškou terénu



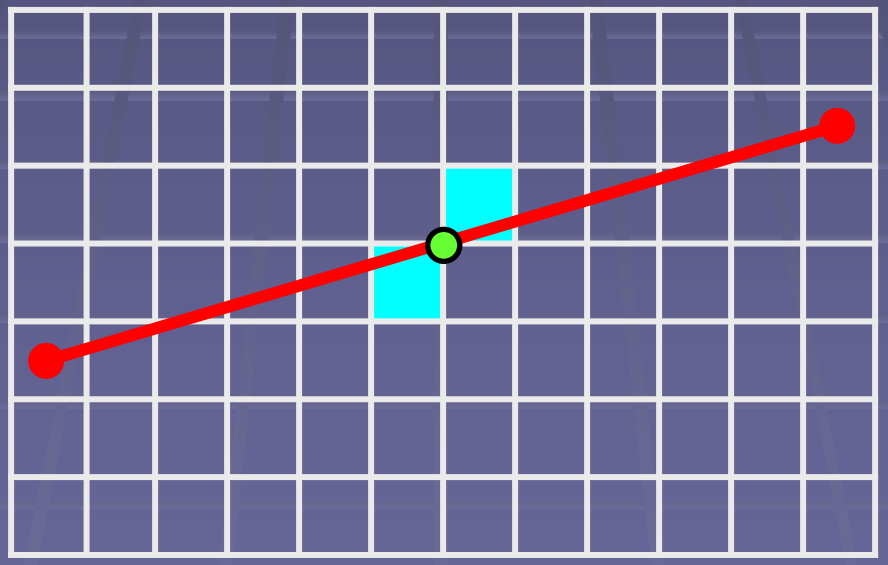
Visible – postup

- Které kostičky protíná úsečka
  - Porovnáme s výškou terénu

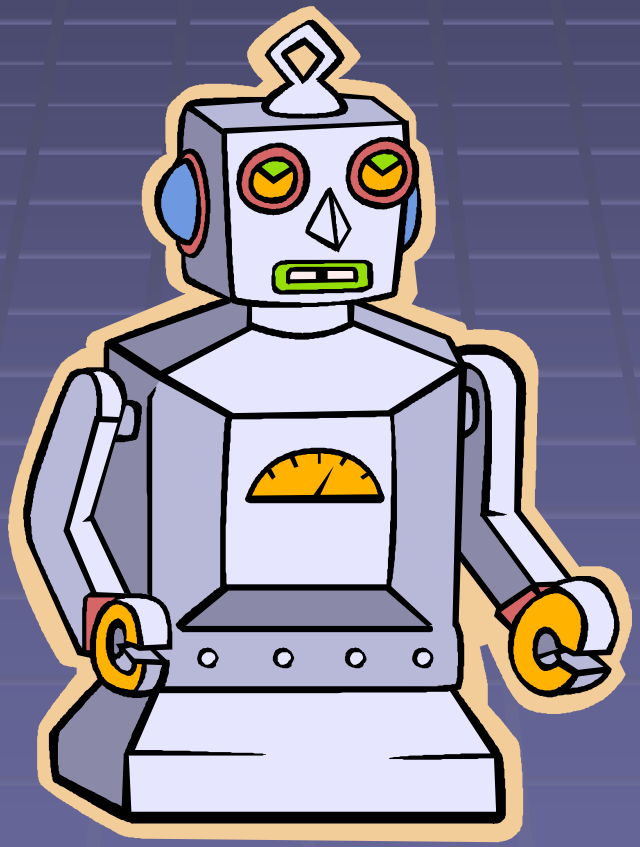


Visible – postup

- Které kostičky protíná úsečka
  - Pozor při přesném průsečíku



KAREL



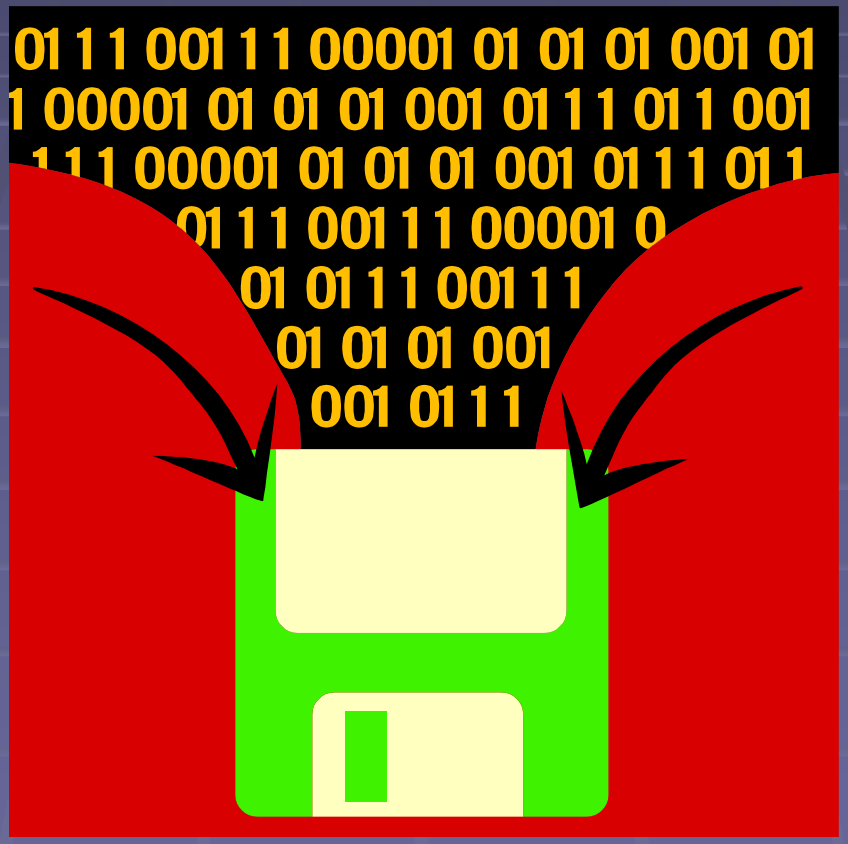
Karel – simulace spuštění

- Spustíme program z každého místa
- Hlídáme, zda jsme se nezacyklili (pamatujeme si stav, kde jsme byli)
  - Pozice
  - Natočení
  - Aktuální příkaz programu

Karel – simulace spuštění

- Při spuštění z nového políčka využijeme dříve vyzkoušené stavy
  - Pozice
  - Natočení
  - Aktuální příkaz programu
    - => maximálně **400 000** možných stavů (stačila i rekurze)

MORE


01 1 1 001 1 1 00001 01 01 01 001 01
1 00001 01 01 01 001 01 1 1 01 1 001
1 1 1 00001 01 01 01 001 01 1 1 01 1
01 1 1 001 1 1 00001 0
01 01 1 1 001 1 1
01 01 01 001
001 01 1 1

More – analogie

- Klasická binární soustava
  - Odzadu nahrazuji jedničky nulami
  - První nulu nahradím 1 a skončím

1 0 1 0 0 1 1 1 1

1 0 1 0 1 0 0 0 0

More – postup

- Negabinární soustava
  - Jen o maličko složitější
  - Střídají se znaménka
    - Změna na správnou stranu => konec
    - Jinak se posunu o pozici dál

1 1 0 1 1 0 1 0 1

1 0 1 0 0 1 0 1 0

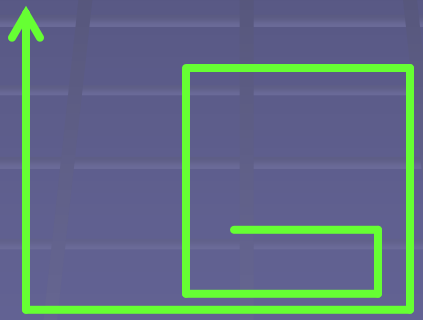
More – kód

```
char c = '0';
for (int i = sb.length()-1; i>=0; --i) {
  char c2 = c=='0'?'1':'0';
  if (sb.charAt(i) == c) {
     sb.setCharAt(i, c2);
     break;
  }
  sb.setCharAt(i, c);
  c = c2;
}
```

SELF

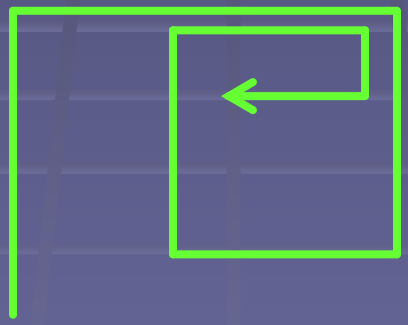
Self – princip

- Jak vypadá platná cesta?
  - **Spirála**



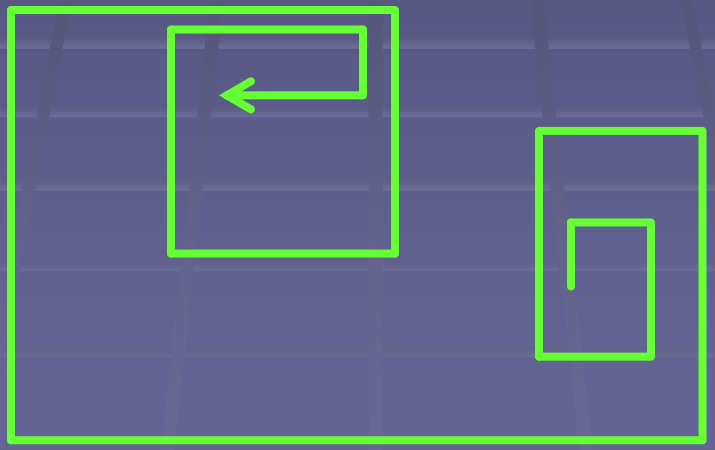
Self – princip

- Jak vypadá platná cesta?
  - Může se ale zatáčet i dovnitř



Self – drobná komplikace

- Jak vypadá platná cesta?
  - Může se to i jednou změnit



Self – zobecnění

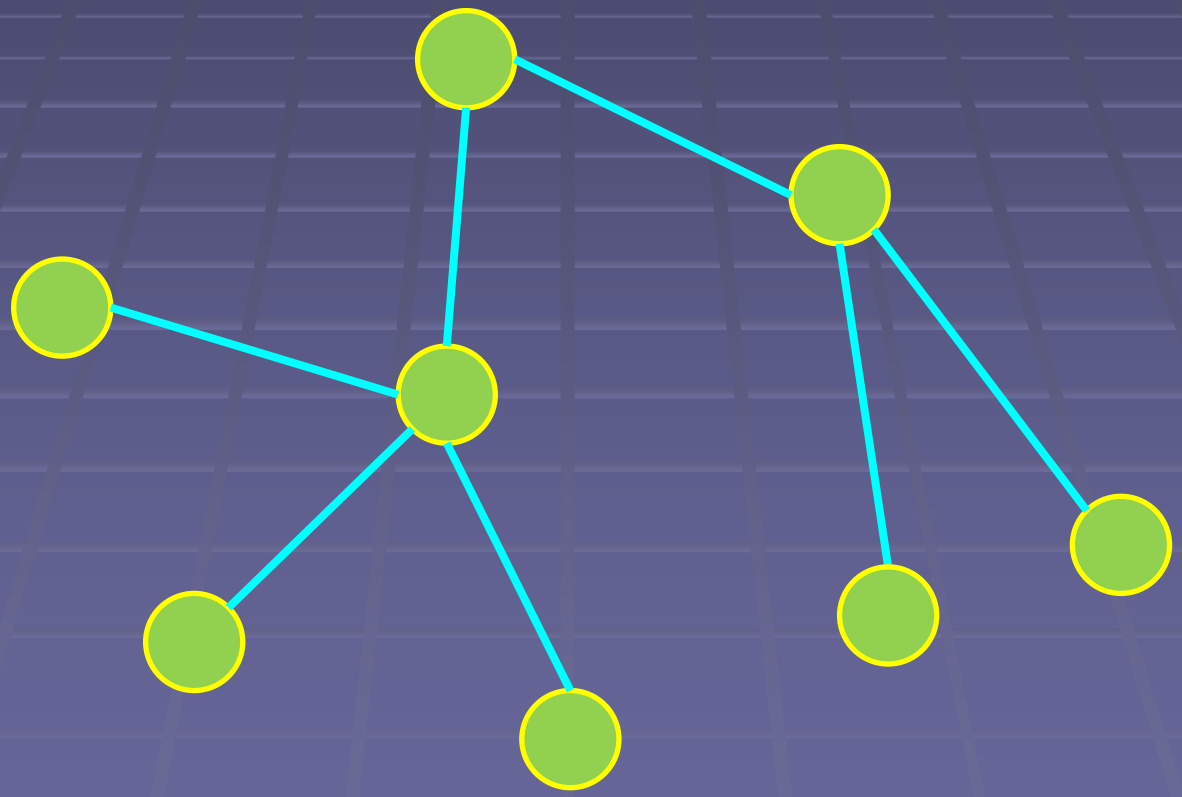
- Část je spirála směrem ven
- Druhá část spirála směrem dovnitř

- => Stačí kontrolovat cca 6 posledních úseků

Self – řešení

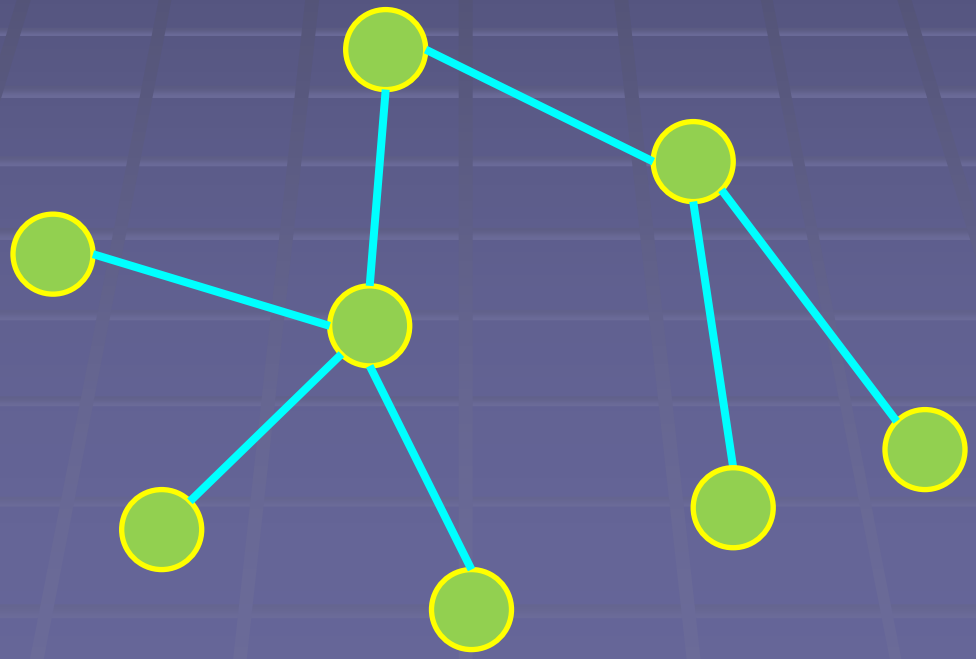
```
while (scanf("%d", &n) == 1) {
   int right0=0, bot0=0, left=0, top=0, right=0, bot=0, norig=n;
   while (n-- > 0) { // out-bound spiral
        scanf("%d", &x); nxt = x - right;
        if (nxt < -right0) { left = nxt; break; }
        else if (nxt <= left) { left = nxt; top = -bot0; break; }
        right0 = bot0; bot0 = left; left = top;
        top = right; right = bot; bot = nxt;
   }
   while (n-- > 0) { // in-bound spiral (inside a rectangle)
        scanf("%d", &x); nxt = x - bot;
        if (nxt >= top) break;
        top = right; right = bot; bot = left; left = nxt;
   }
   printf((n < 0) ? "OK\n" : "%d\n", norig-n-1);
   while (n-- > 0) scanf("%d", &x);
}
```

LENGTHY



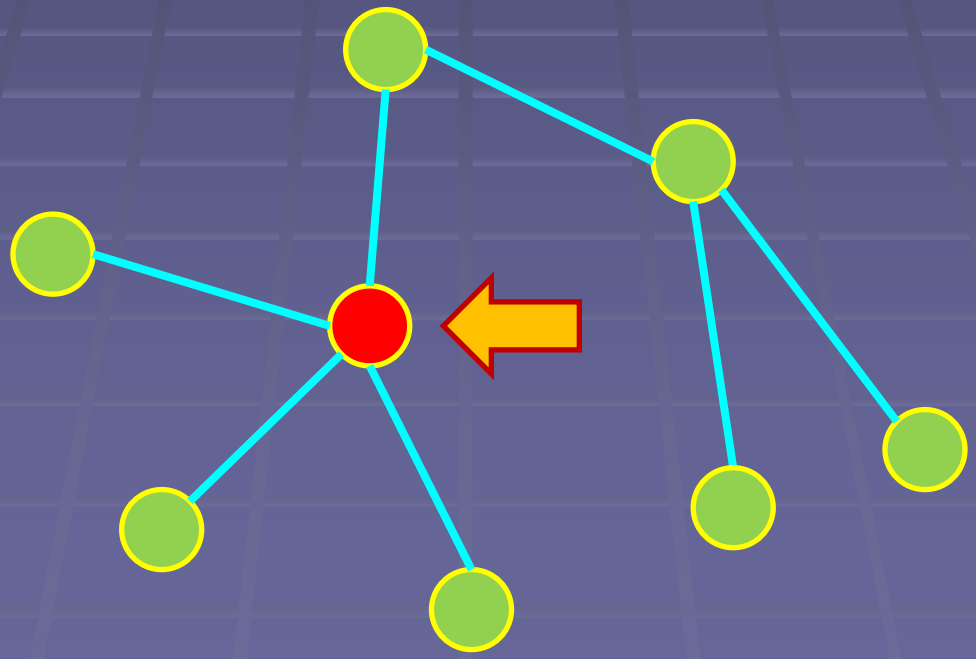
Lengthy – zadání

- Permutace uzlů stromu s nejdelší celkovou (uzavřenou) cestou



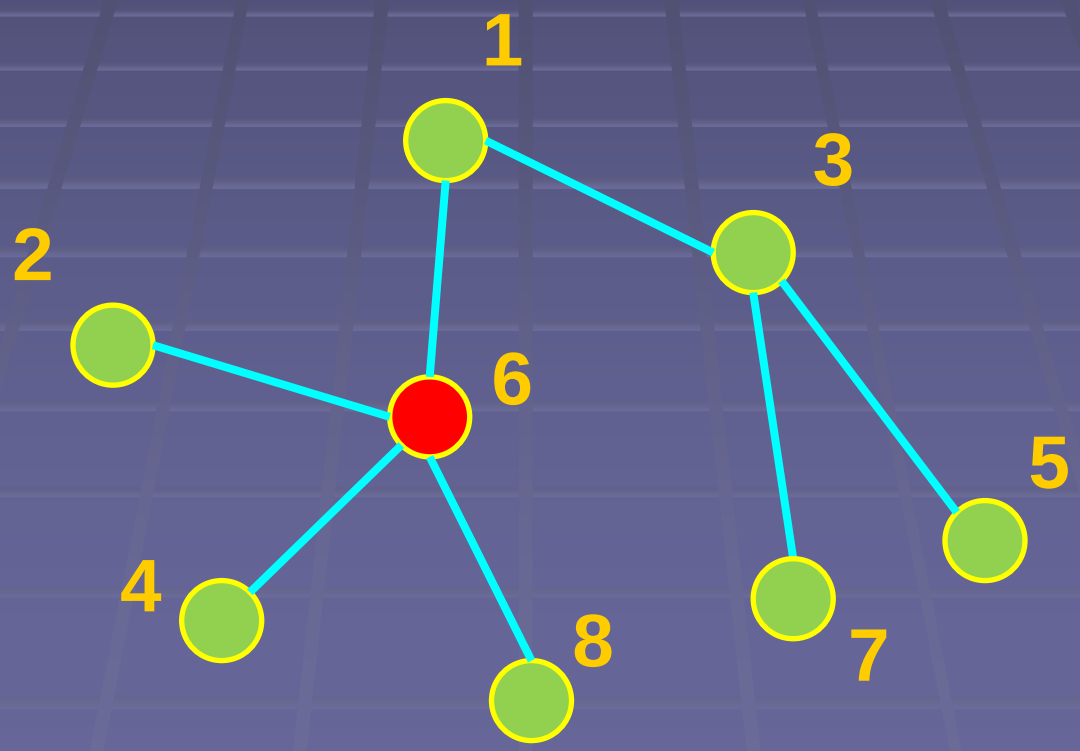
Lengthy – princip

- Nalezneme „těžiště“ stromu (uzel, jehož žádný podstrom nemá více než ½ všech uzlů)



Lengthy – princip

- Pak už stačí střídat podstromy
  - Použít lze i uzel sám

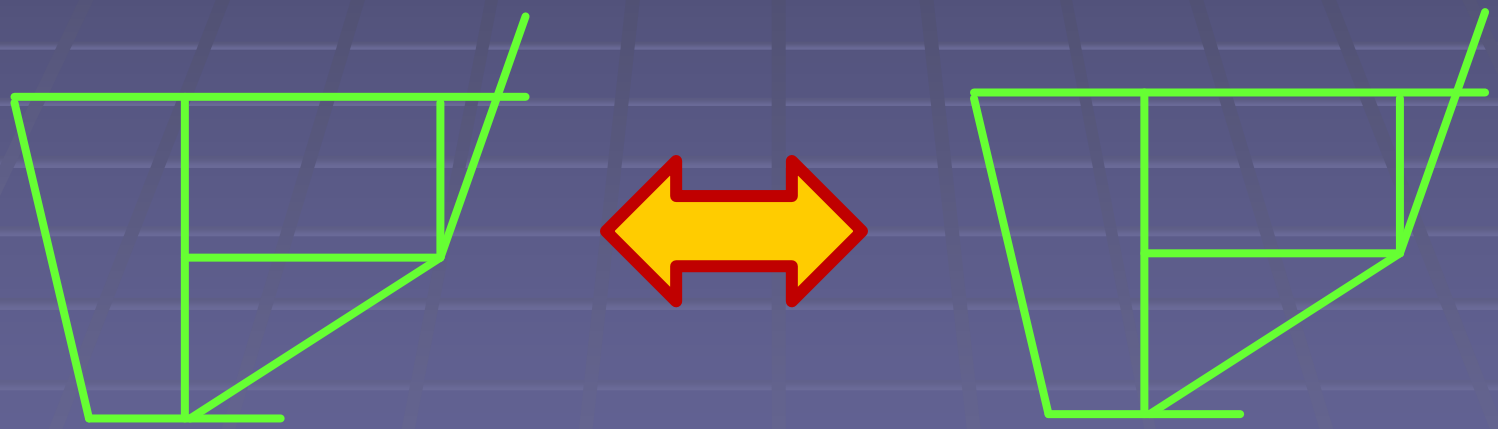


Lengthy – proč to funguje

- Každá cesta jde přes „těžiště“
  - => Každá hrana se použije tolikrát, kolik je uzlů na její „menší“ straně
- Více to nejde

(každé řešení vypadá takto)

VALIDATE



Validate

- Myšlenka není těžká
- Obtížnější implementace

Validate – postup

- Rozdělení úseček podle směru (a pozice)
- V každém směru sloučit překryvy
  - Seřazení a spojení intervalů
  - .....

DOTAZY

Autoři úloh

Josef Cibulka

Michal „Mimino“ Danilák

Zdeněk Dvořák

Martin Kačer

… a otcové zakladatelé ☺

A tradičně náměty: Radek Pelánek