2. Mřížky / Záplavové vyplňování

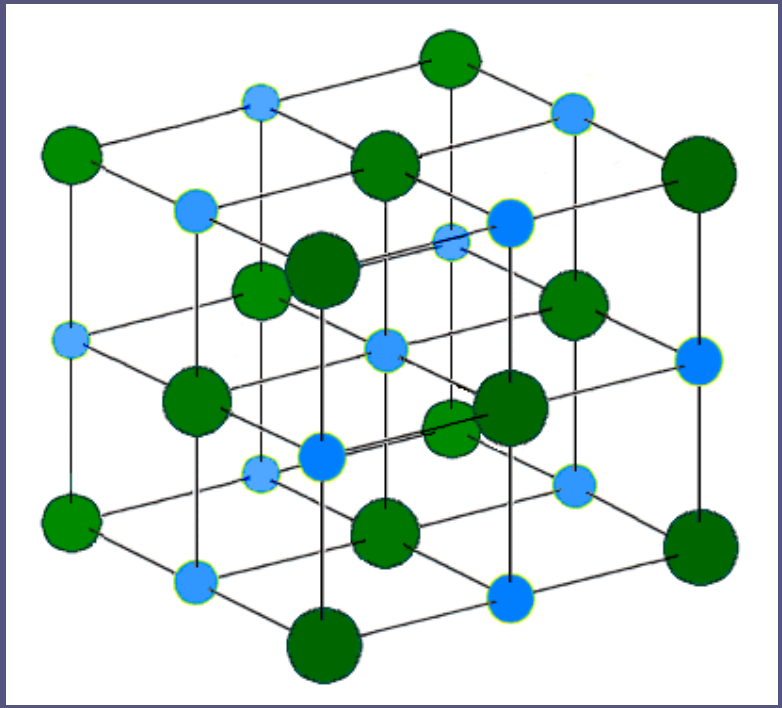
BI-EP2
Efektivní programování 2

LS 2025/2026
Ing. Martin Kačer, Ph.D.

Katedra teoretické informatiky
Fakulta informačních technologií
České vysoké učení technické v Praze

Mřížky

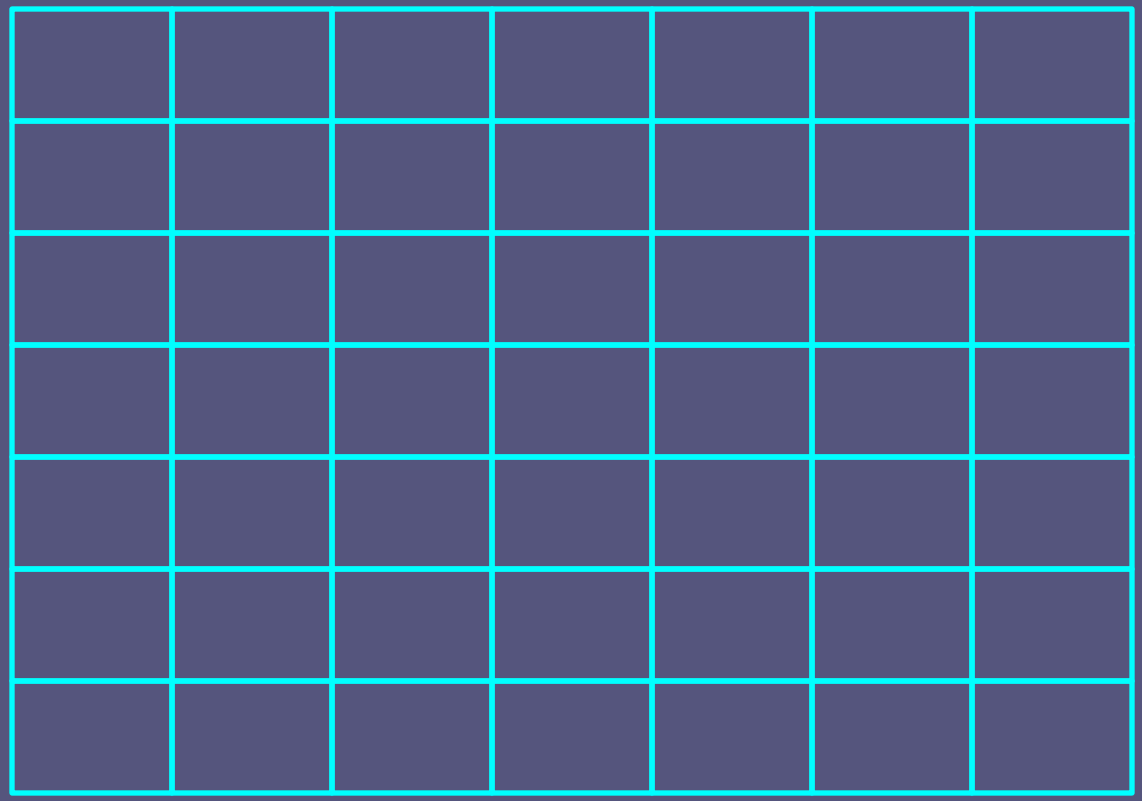
- Pravidelná struktura
- Anglický název
  - Grid ←
  - Lattice
  - Raster
  - Mesh



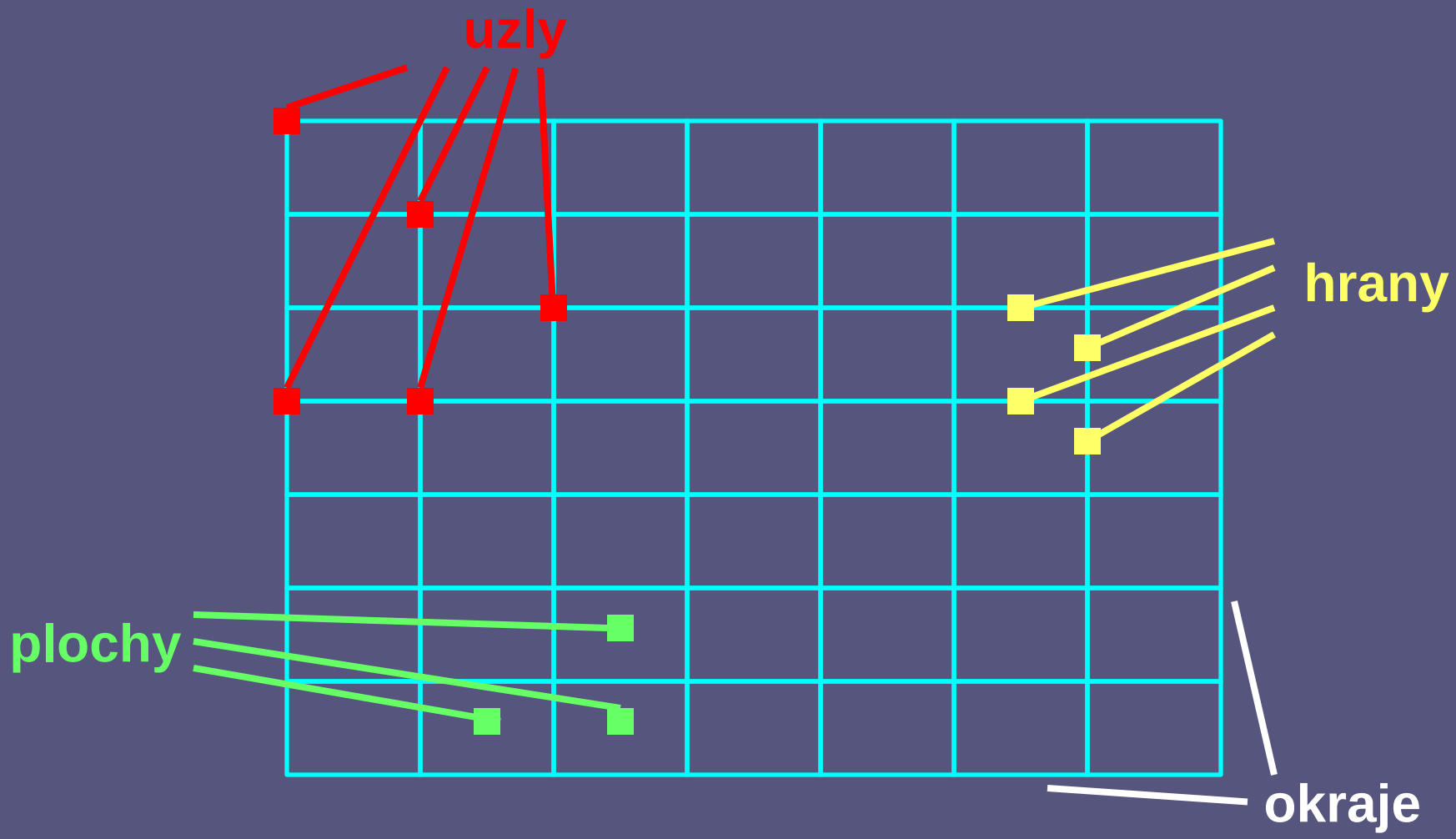
Význam mřížek

- Pravidelné rozdělení prostoru
  - Modelování a reprezentace
- Části mřížky
  - Uzly
  - Hrany
  - „Vnitřky“ (Plochy, Stěny)

Pravoúhlé mřížky



Pravoúhlé mřížky



Pravoúhlé mřížky – reprezentace

- Reprezentace pomocí matice (2D pole)
  - Přímočará
  - Snadná

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7

Pravoúhlé mřížky – data

- Co do matice ukládat?
  - Záleží na úloze…
  - Příklad: Stav políčka – plné/prázdné

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7

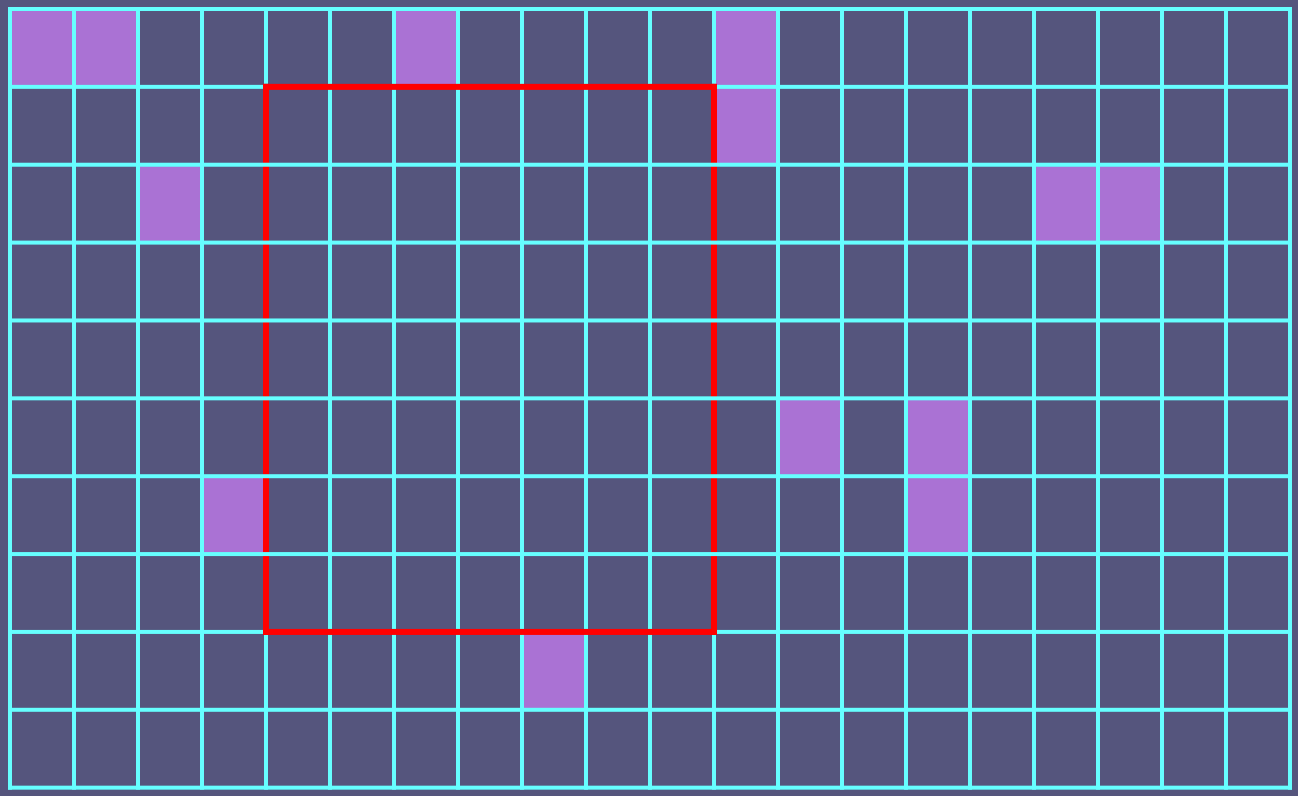
Pravoúhlé mřížky

- Co do matice ukládat?
  - Záleží na úloze…
  - Příklad: Barva nebo odstín

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7

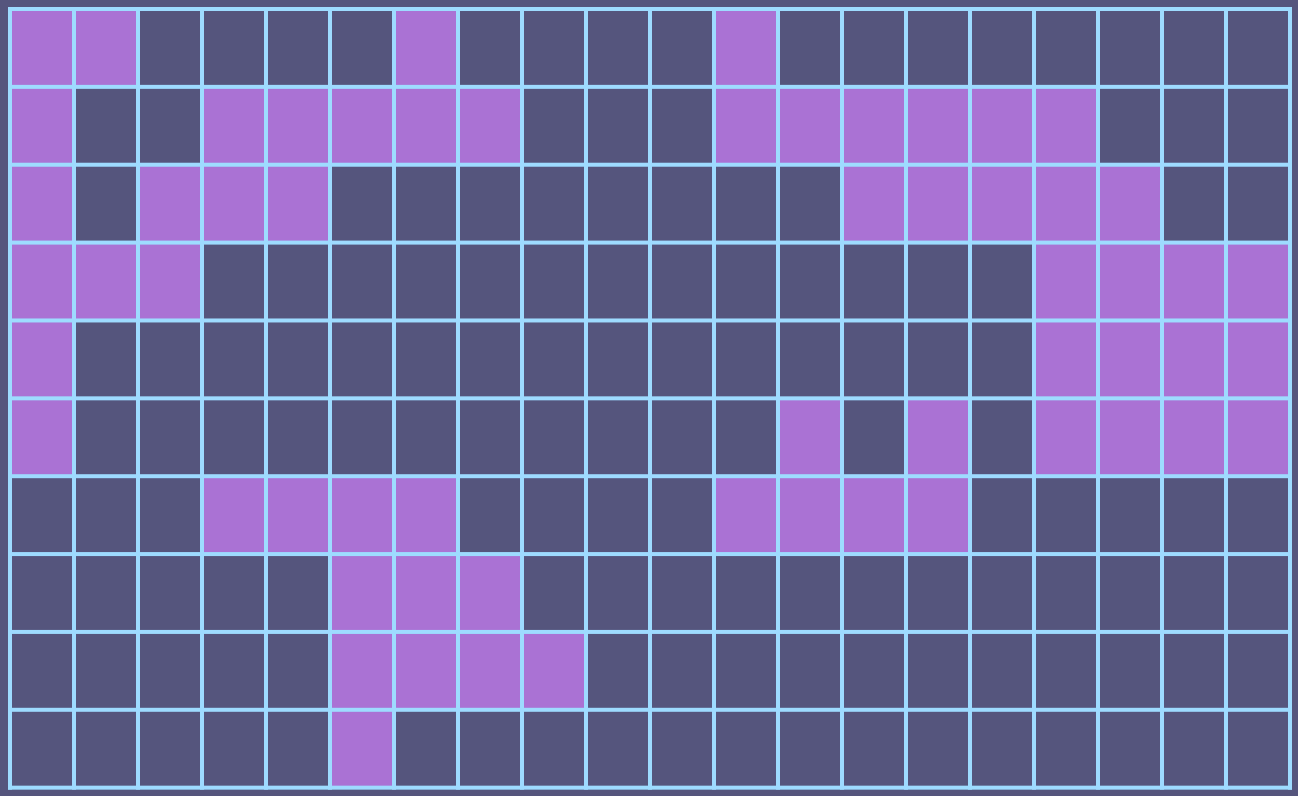
Příklady úloh s mřížkou

- Maximální prázdný/plný obdélník



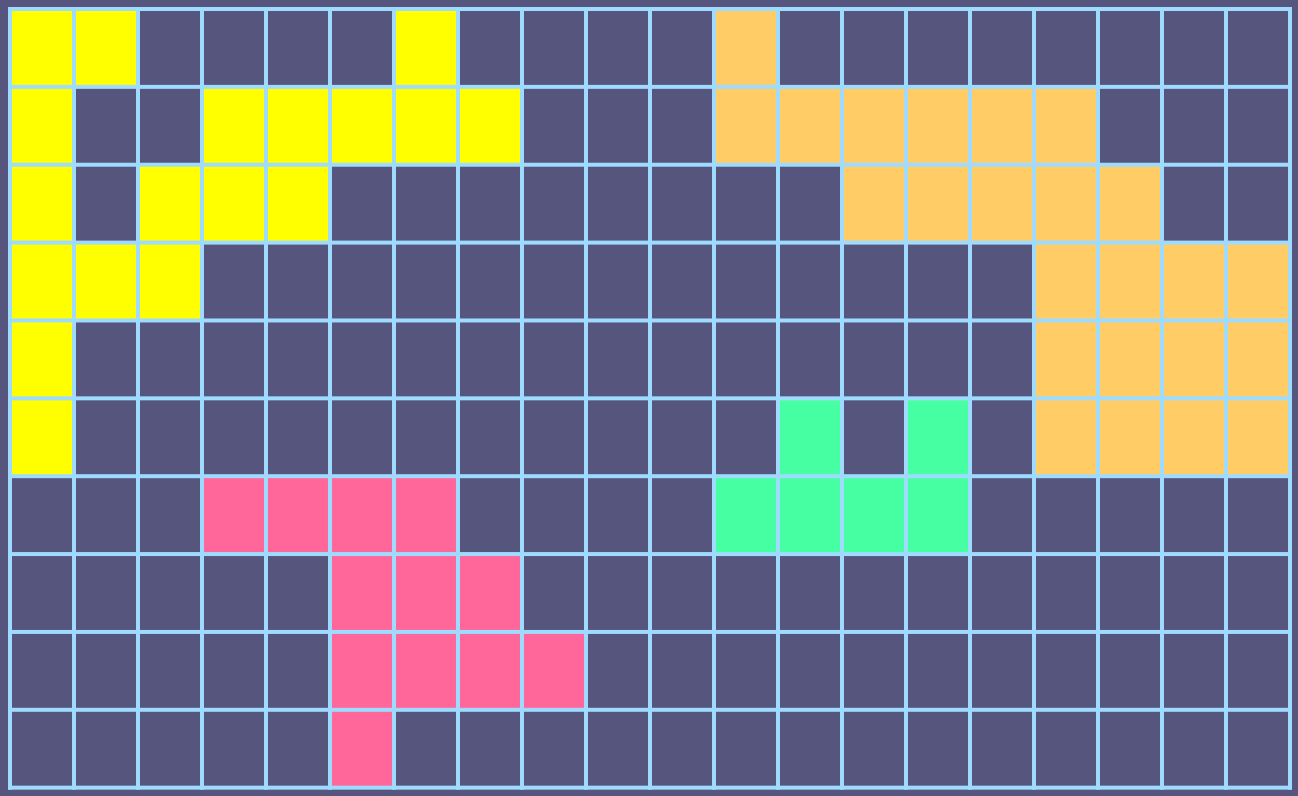
Příklady úloh s mřížkou

- Detekce souvislých oblastí



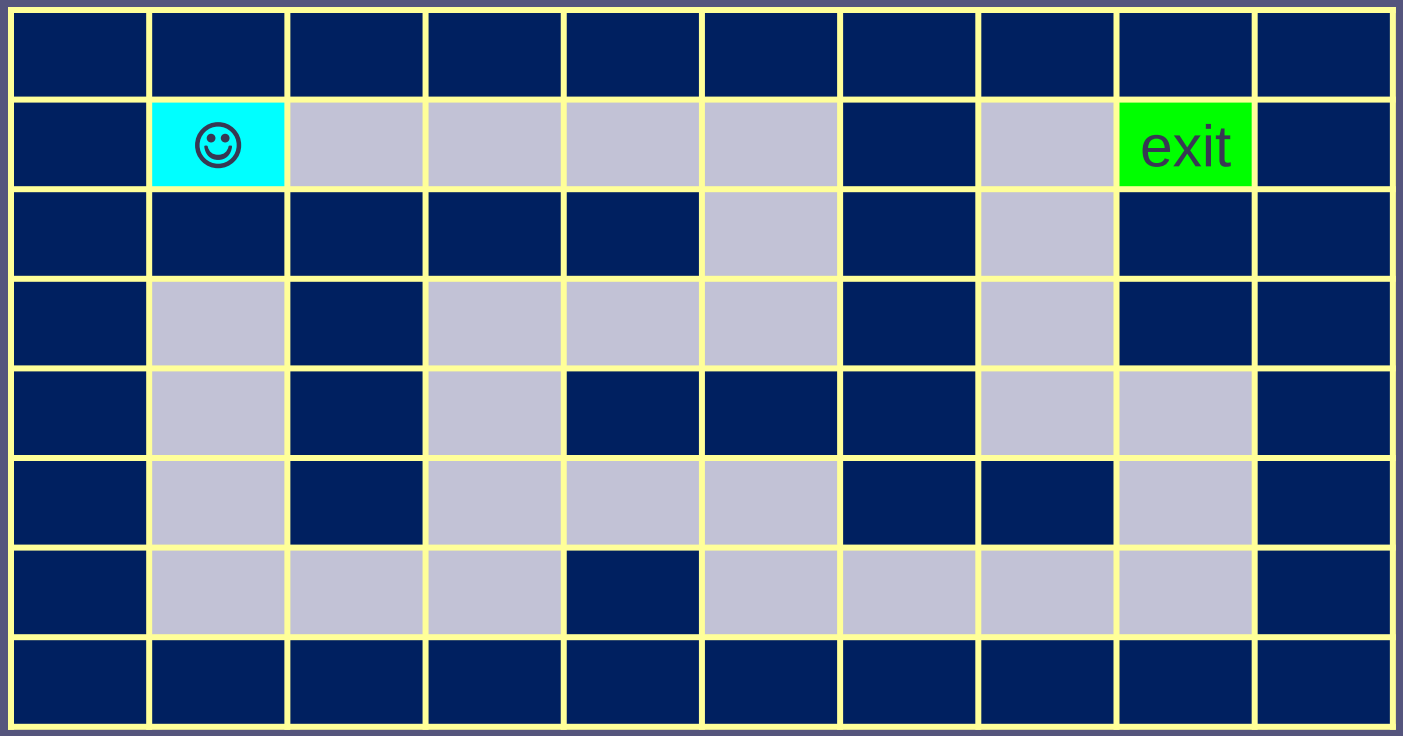
Příklady úloh s mřížkou

- Detekce souvislých oblastí



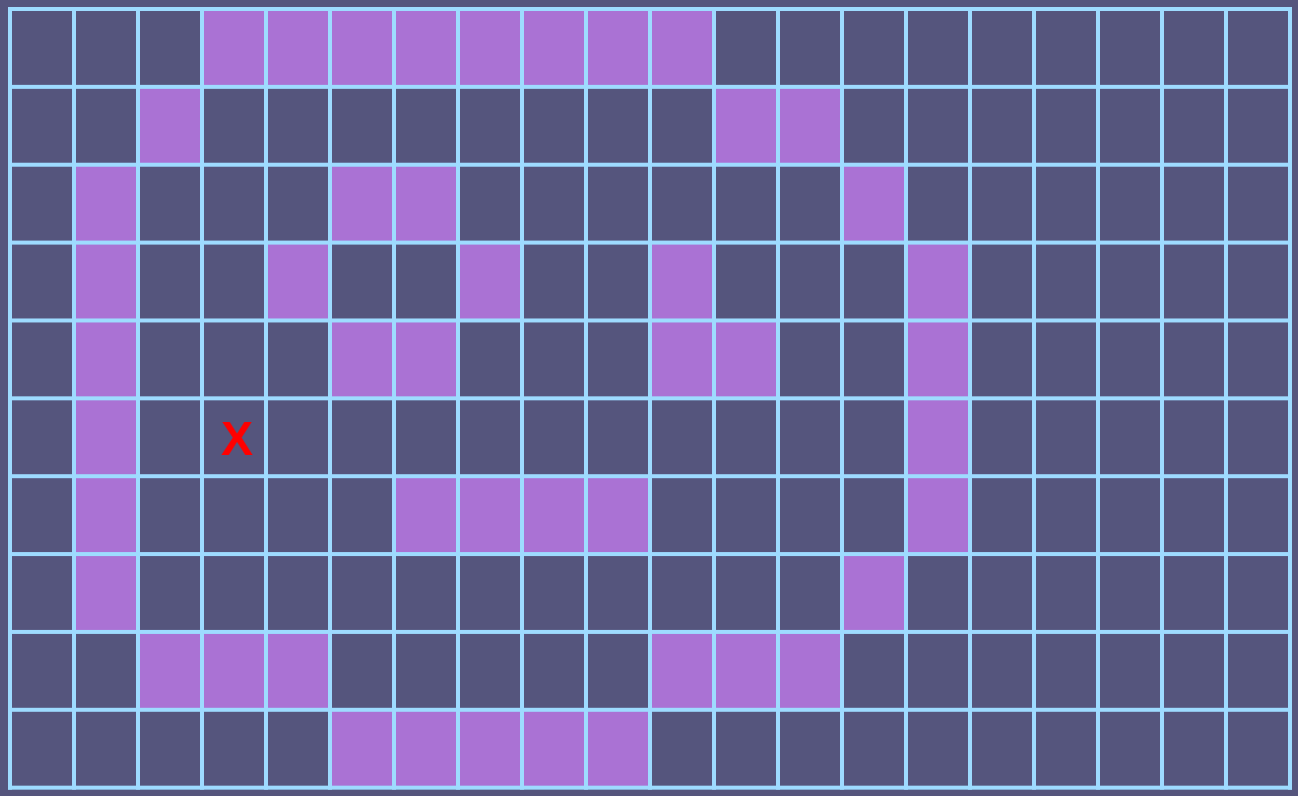
Příklady úloh s mřížkou

- Průchod bludištěm
  - Různě komplikovaný (klíče atp.)



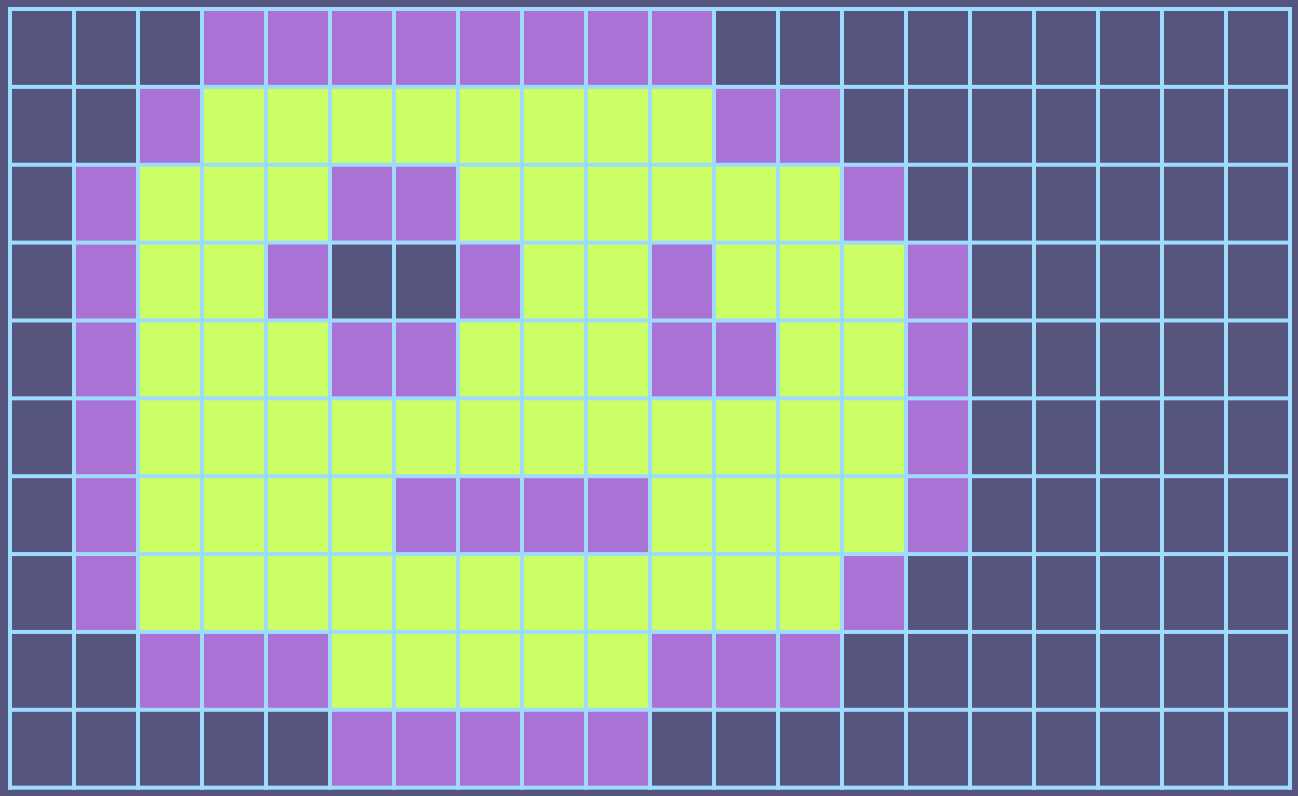
Příklady úloh s mřížkou

- Vyplnění obrysu

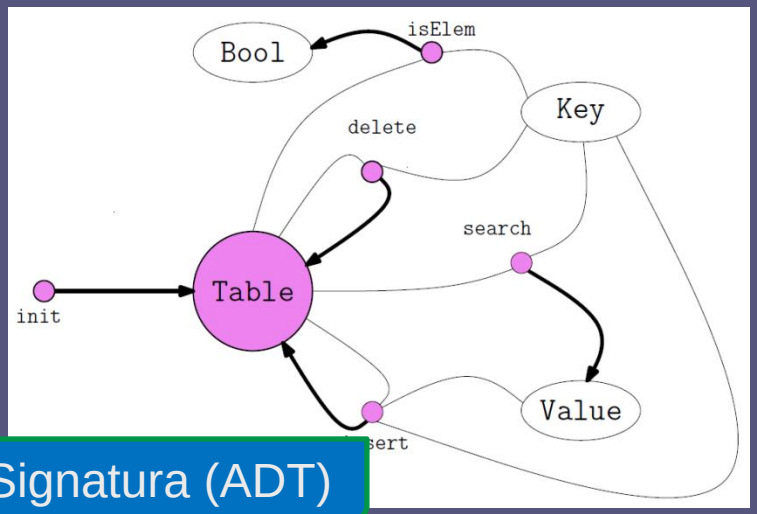


Příklady úloh s mřížkou

- Vyplnění obrysu



„Nebojte se datových struktur“



Signatura (ADT)

- Časová a paměťová složitost
- Podmínky & další omezení
- … případně další

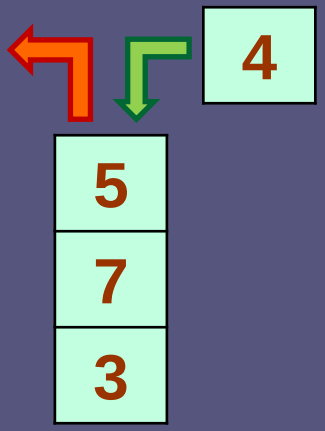
Vlastnosti implementace

```
function Vlož_prvek(table, x) {
    Hash h = spočítej_hash(x);
    if (existuje(h)) {
          h = další_hash(h, x);
    }
    tabulka_vlož(h, x)
}
```

Pseudokód

```
public V put(K key, V value){
    int h = hash(key);
    . . .
    if (tab[i =(n-1)&hash])
          tab[i] = ...;
    else {
       if (p.hash == hash &&
             (p.key == key ||
                key=k))
          e = p;
       . . .
    }
    return oldValue
}
```

Kód



Příklad: Zásobník

- Init
- Push
- Pop
- Is Empty

```
int stack[MAX], top;

top = 0;

stack[top++] = x;

return stack[--top];

return !top;
```

Příklad: Fronta

- Init
- Enqueue
- Dequeue
- Is Empty

```
int q[MAX], qh, qt;

qh = qt = 0;

q[qt++ % MAX] = x;

return q[qh++ % MAX];

return qh == qt;
```

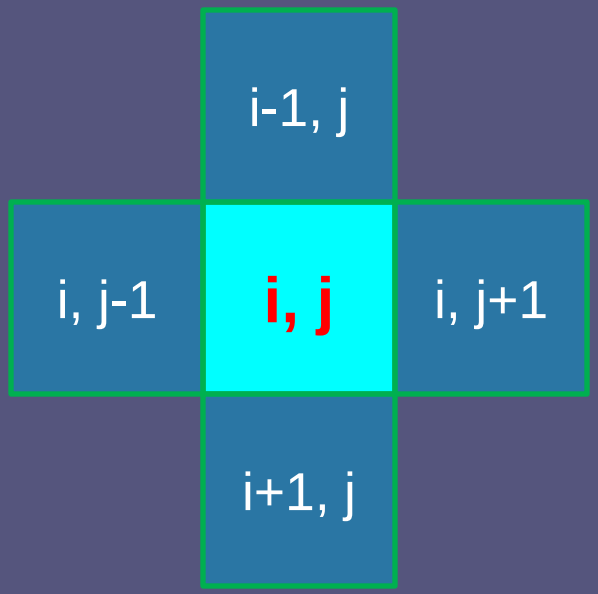
Operace s mřížkou

Jaké operace potřebujeme?

- Přístup k danému poli
- Zjištění sousedních polí
- Rozpoznání okrajů
- Projití celé oblasti (iterace)
  - Různé pořadí (po řádcích, diagonálně atp.)

Operace – pravoúhlé mřížky

- Sousední pole jsou jednoduchá



Operace – pravoúhlé mřížky

- Okraje – záleží na tvaru
- V případě obdélníku (čtverce) opět snadné

i == 0

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	**i, j**	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7

j == 0

j == COLS-1

i == ROWS-1

Iterace – pravoúhlé mřížky

- Po řádcích (row-major)

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

```
for (int i = 0; i < ROWS; ++i)
  for (int j = 0; j < COLS; ++j) {
     do_something(i, j);
  }
```

Iterace – pravoúhlé mřížky

- Po sloupcích (column-major)

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

```
for (int j = 0; j < COLS; ++j)
  for (int i = 0; i < ROWS; ++i) {
     do_something(i, j);
  }
```

Iterace – pravoúhlé mřížky

- Hadovitě (snake-like)

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

```
for (int i = 0; i < ROWS; i += 2) {
  for (int j = 0; j < COLS; ++j) {
     do_something(i, j);
  }
  if (i+1 < ROWS)
     for (int j = COLS; --j >= 0; ) {
          do_something(i+1, j);
}    }
```

Iterace – pravoúhlé mřížky

- Hadovitě (snake-like)
  - Jiná možnost

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

```
for (int i = 0; i < ROWS; ++i)
  for (int j = 0; j < COLS; ++j) {
     do_something(i,
          j + (COLS-1-2*j) * (i%2));
  }
```

Iterace – pravoúhlé mřížky

- Diagonálně (diagonal)

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

```
????
```

Zarážky na okraji

- Může se vyplatit přidat na kraj pole navíc
  - Zjednodušuje podmínky

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7

Zarážky na okraji

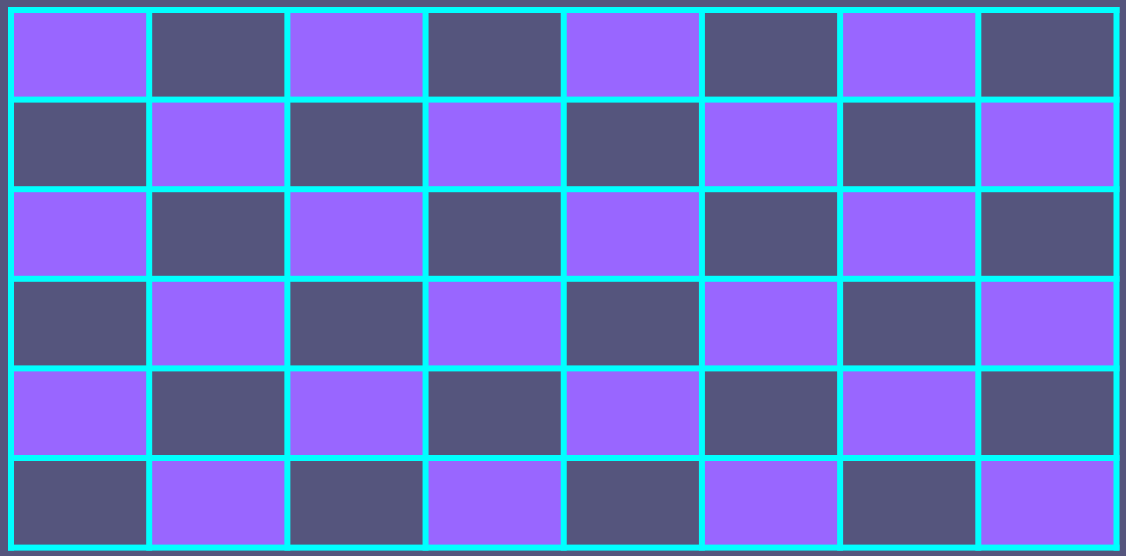
- Může se vyplatit přidat na kraj pole navíc

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8	5,9
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8	6,9
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8	7,9

Méně pravidelné mřížky

- Cokoli, co není pravidelný obdélník
  - Chybějící políčka
  - Nepravidelná struktura
  - Nepravoúhlá povaha
  - …
- Potřeba najít vhodné indexování
  - Viz operace: sousedé, okraje, iterace

Šachovnice



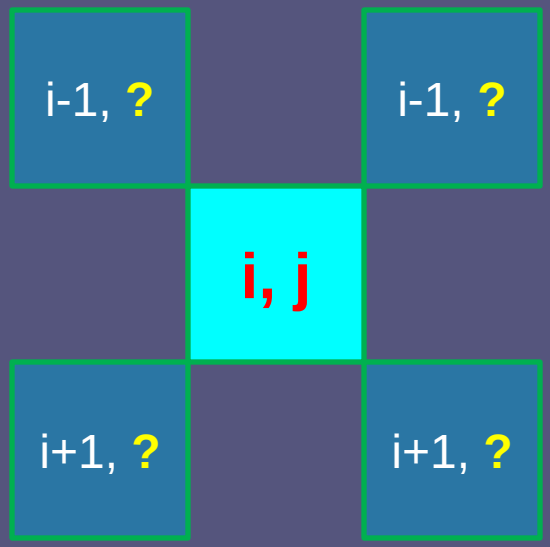
Šachovnice – varianta 1

0,0		0,1		0,2		0,3	
	1,0		1,1		1,2		1,3
2,0		2,1		2,2		2,3	
	3,0		3,1		3,2		3,3
4,0		4,1		4,2		4,3	
	5,0		5,1		5,2		5,3

- Okraje dobře definované
- Iterace také poměrně jednoduchá
  - … jen pozor při lichém počtu sloupců!

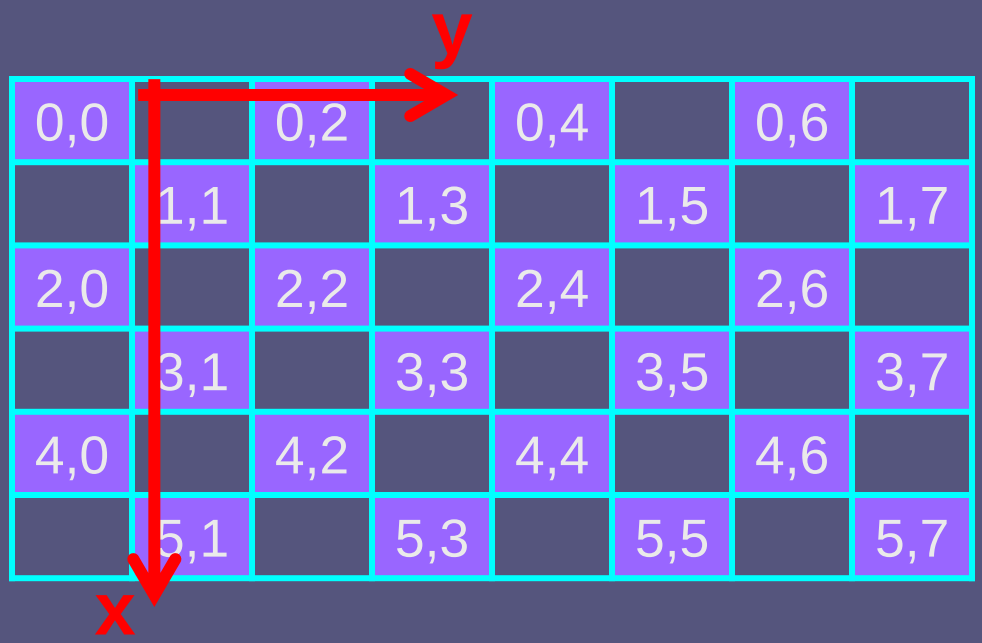
Šachovnice – varianta 1

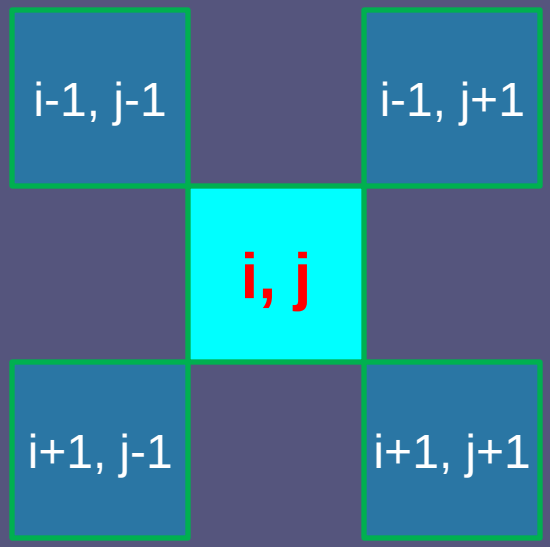
0,0		0,1		0,2		0,3	
	1,0		1,1		1,2		1,3
2,0		2,1		2,2		2,3	
	3,0		3,1		3,2		3,3
4,0		4,1		4,2		4,3	
	5,0		5,1		5,2		5,3



- Ale netriviální zjištění sousedů!
  - Rozdíl: lichý x sudý řádek

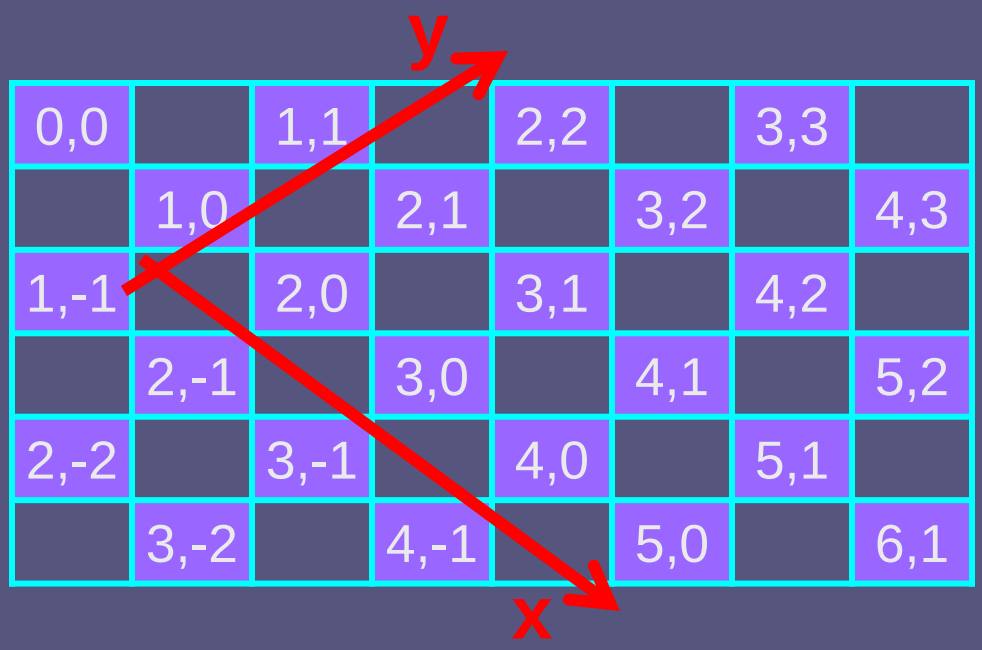
Šachovnice – varianta 2

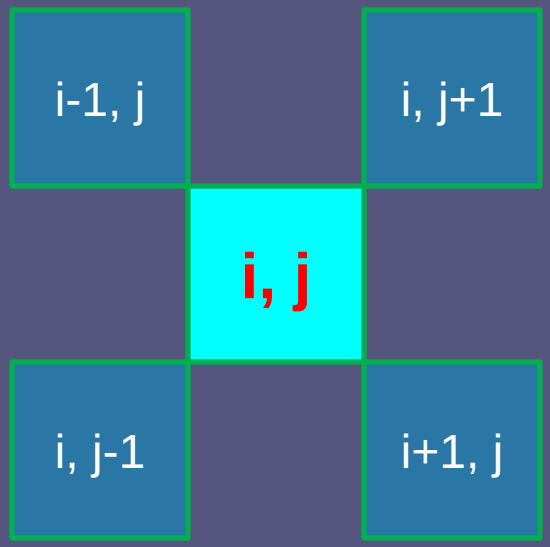




- Iterace i okraje stále dobré
- Dokonce i sousedé
- Ale pozor – plýtváme pamětí!

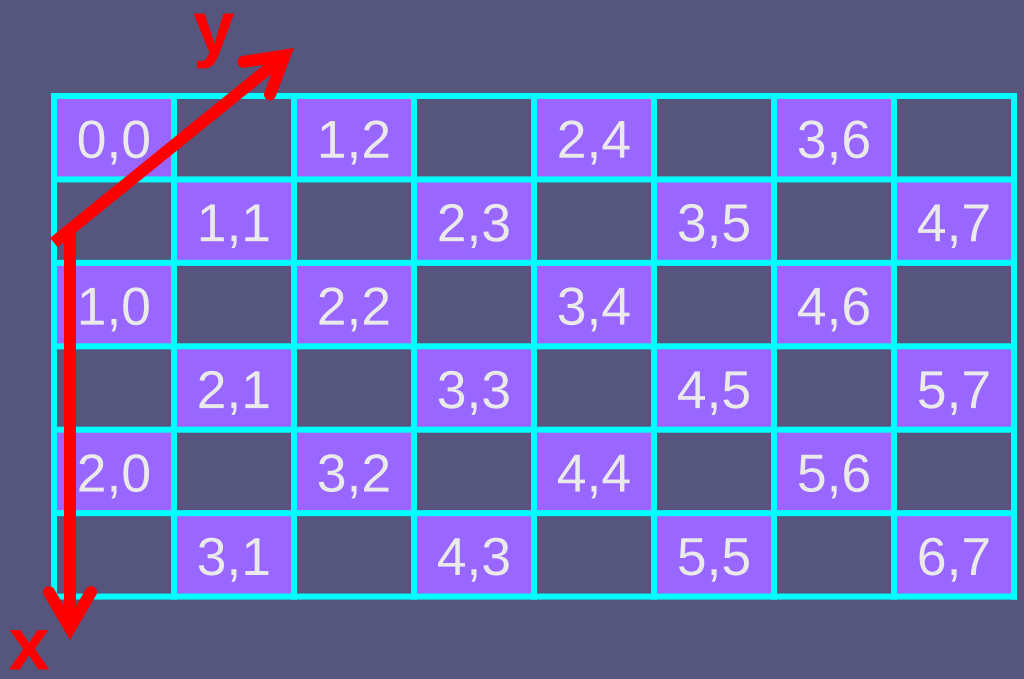
Šachovnice – varianta 3

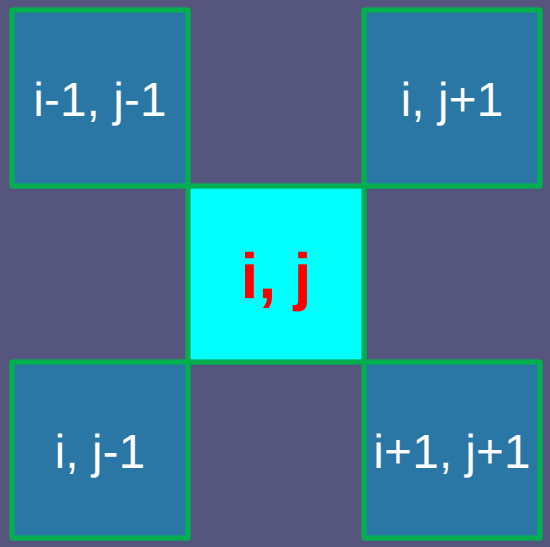




- Sousedé ok
- Složitější okraje a iterace
- A navíc záporné indexy…

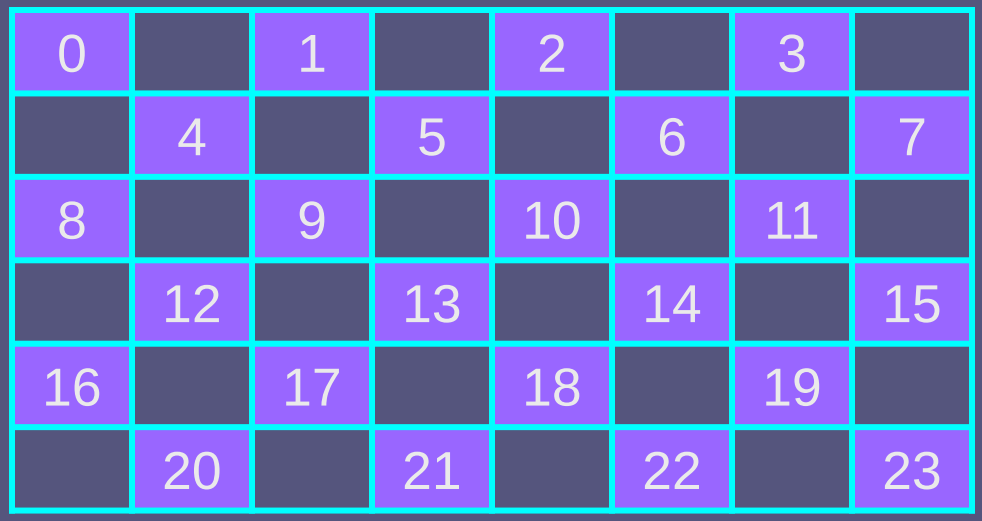
Šachovnice – varianta 4

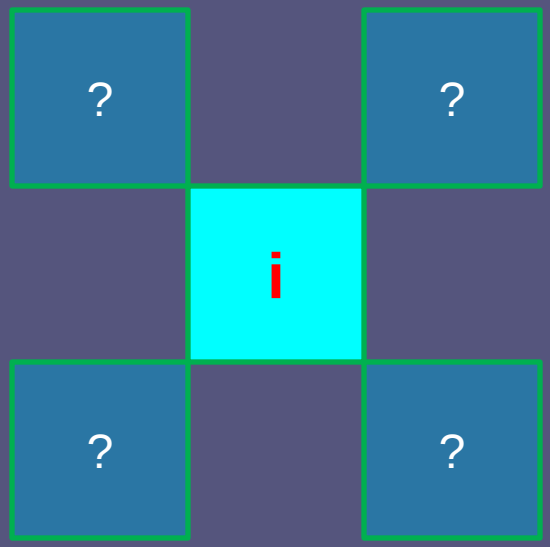




- Sousedé ok
- Složitější okraje a iterace
- … a už zase plýtváme (i když trochu jinak)

Šachovnice – varianta 5



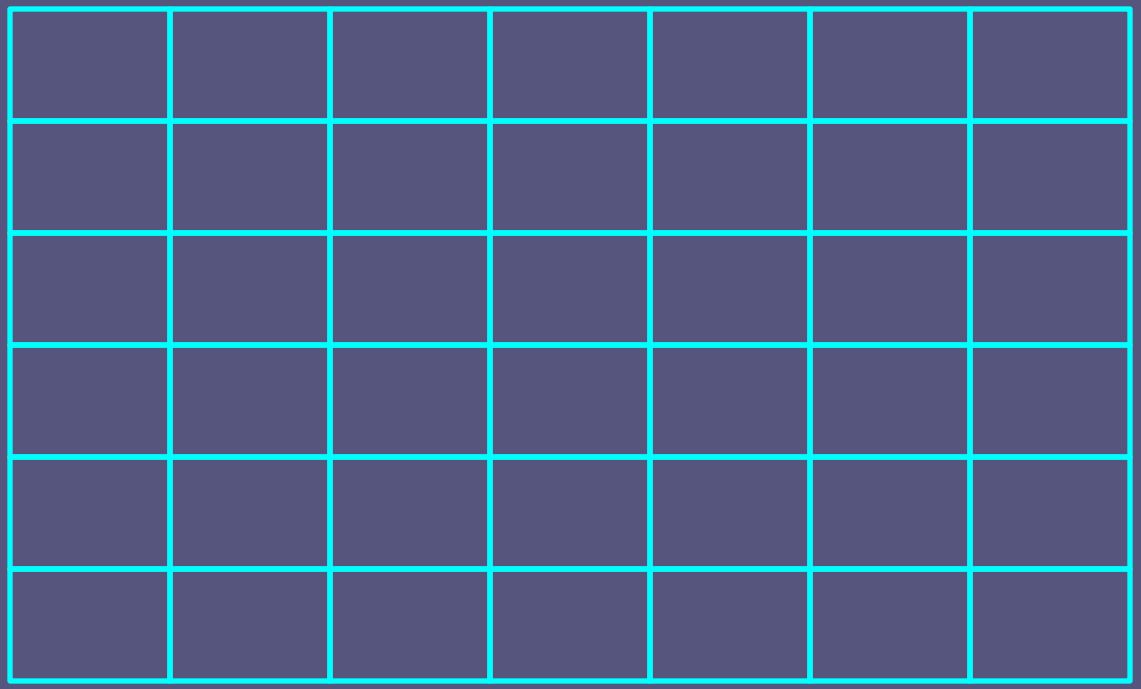


Vlastní mapovací funkce

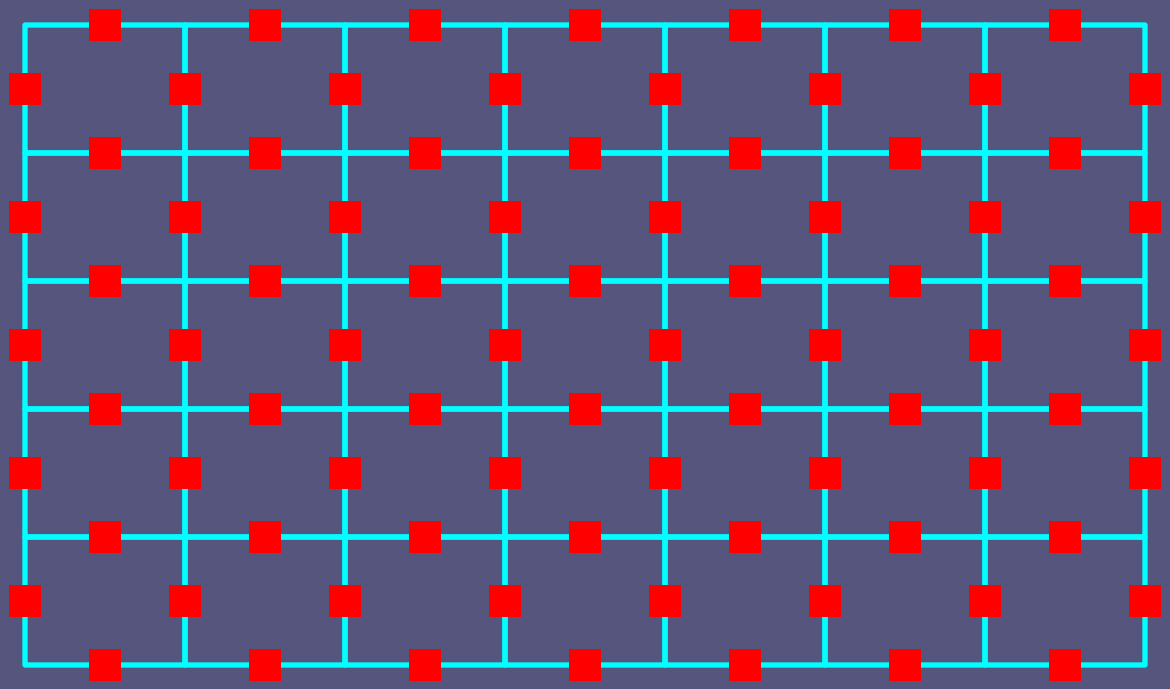
- Iterace je přímočará
- Složitější okraje a sousedé

Hrany mezi čtverečky

- Zajímají nás hrany, nikoli políčka



Hrany mezi čtverečky



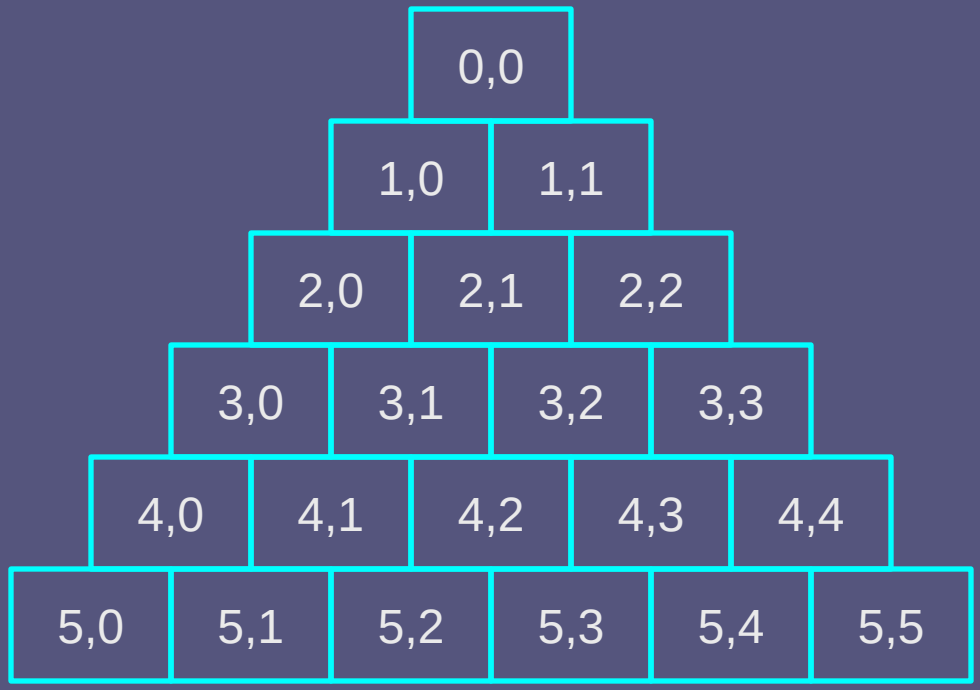
- Trochu podobné jako „šachovnice“

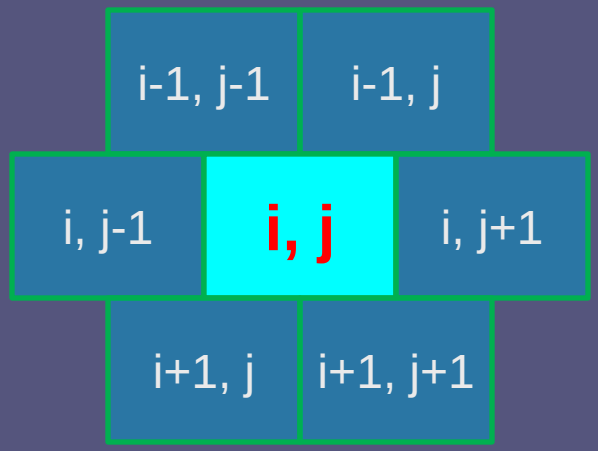
Hrany mezi čtverečky

	0,0	0,1	0,2	0,3	0,4	0,5
0,0	0,1 1,0	0,2 1,1	0,3 1,2	0,4 1,3	0,5 1,4	0,6 1,5
1,0	1,1 2,0	1,2 2,1	1,3 2,2	1,4 2,3	1,5 2,4	1,6 2,5
2,0	2,1 3,0	2,2 3,1	2,3 3,2	2,4 3,3	2,5 3,4	2,6 3,5
3,0	3,1 4,0	3,2 4,1	3,3 4,2	3,4 4,3	3,5 4,4	3,6 4,5

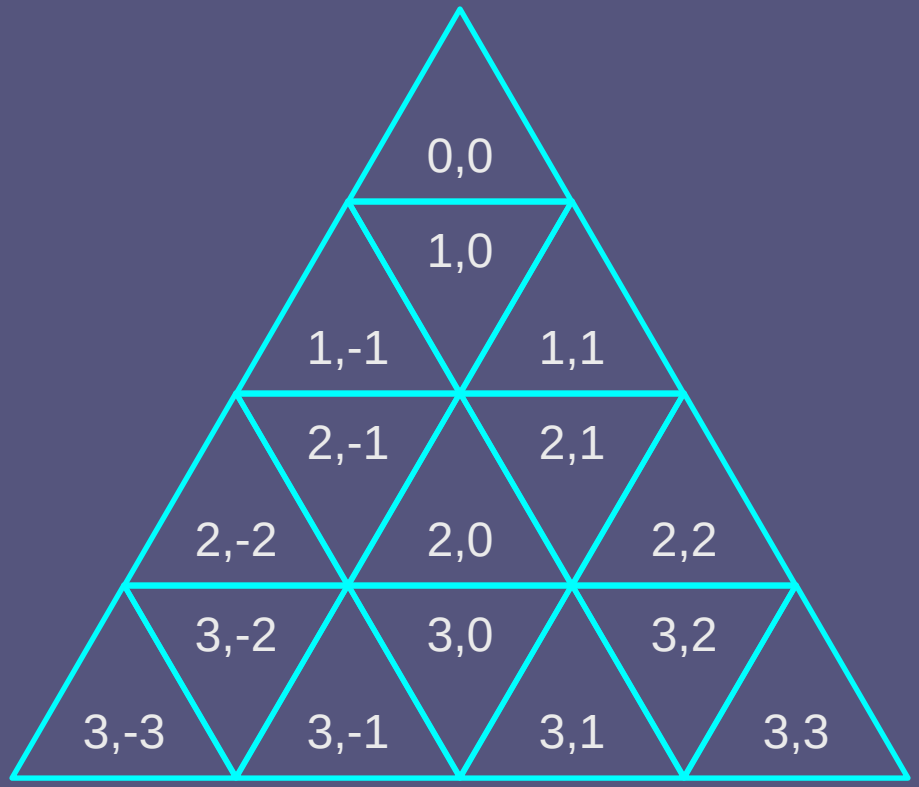
- Lépe 2 matice: **vodorovné** [R+1][C]
  svislé [R][C+1]

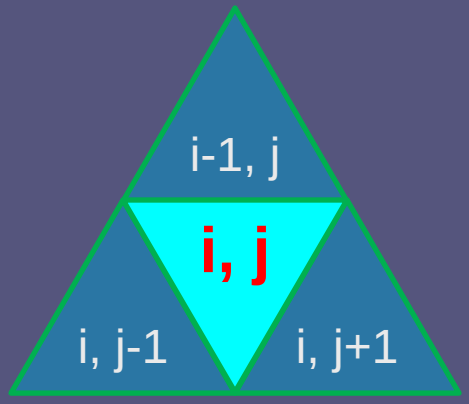
Trojúhelníkové mřížky

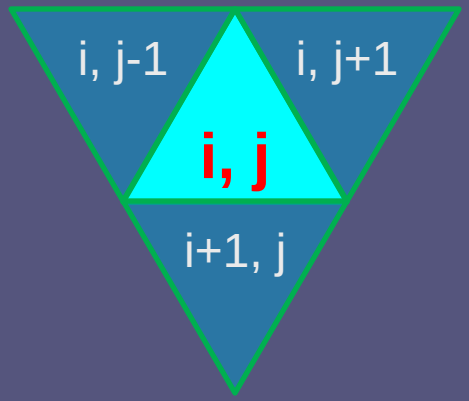




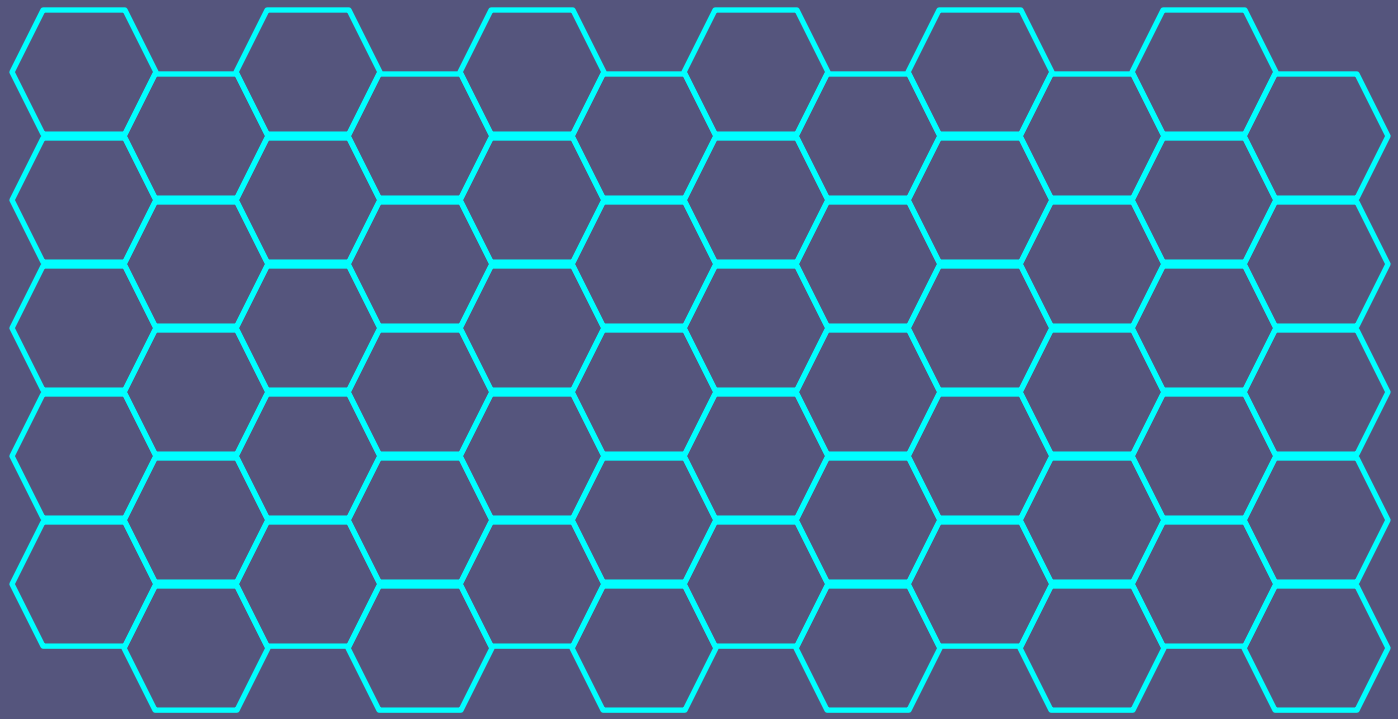
Trojúhelníkové mřížky



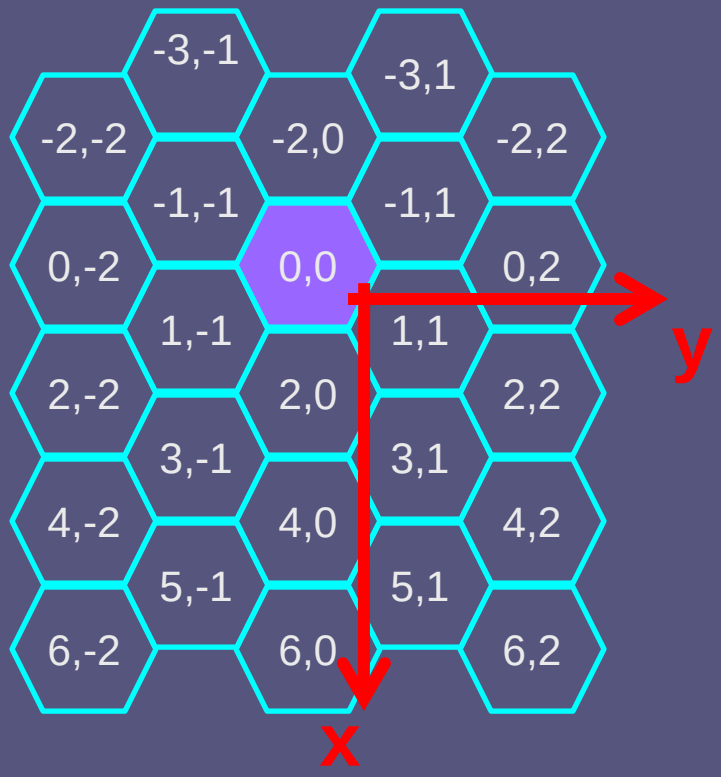


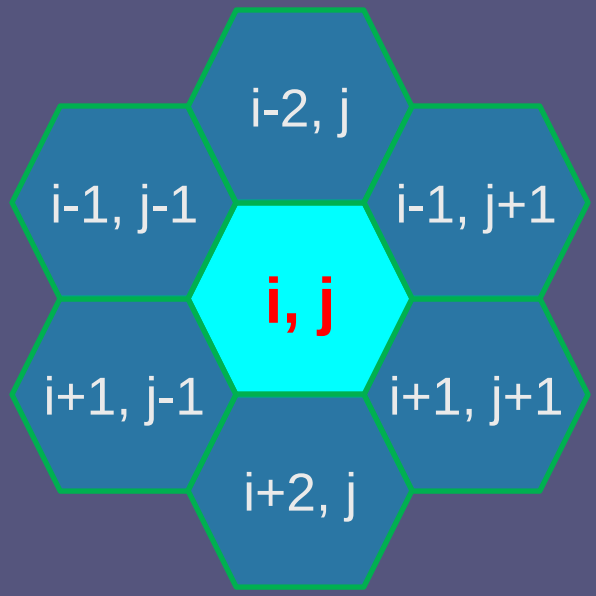


Šestiúhelníkové mřížky

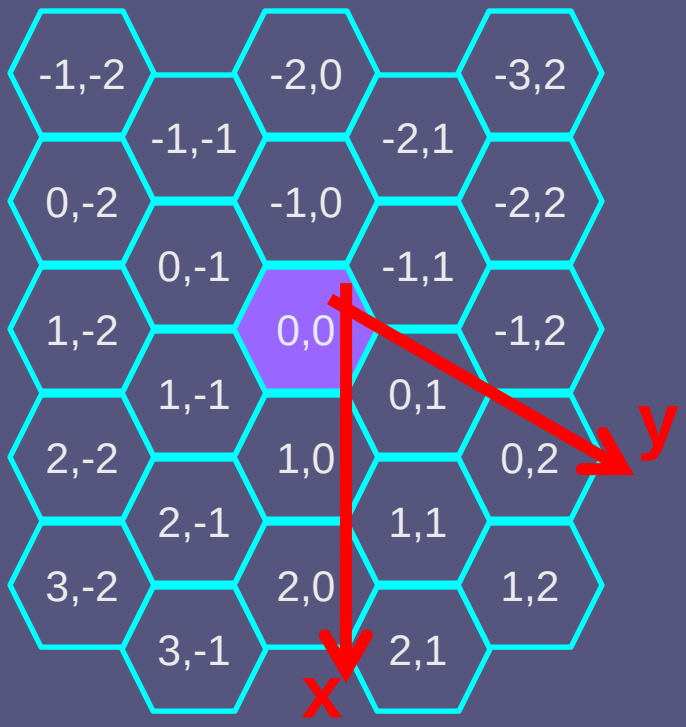


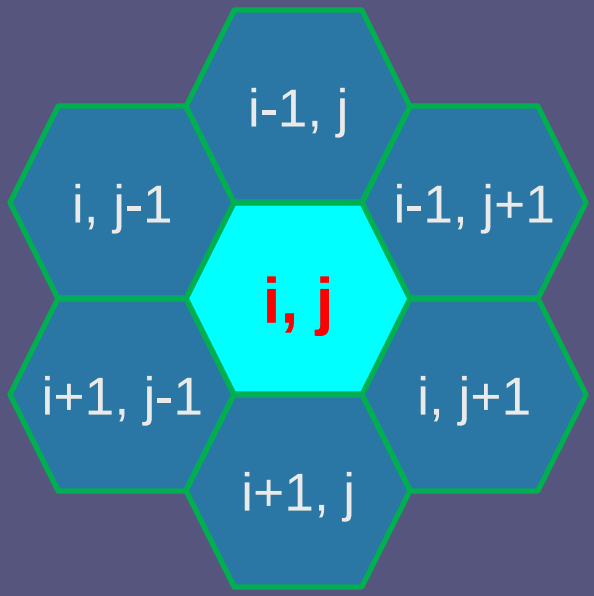
Šestiúhelníkové mřížky





Šestiúhelníkové mřížky



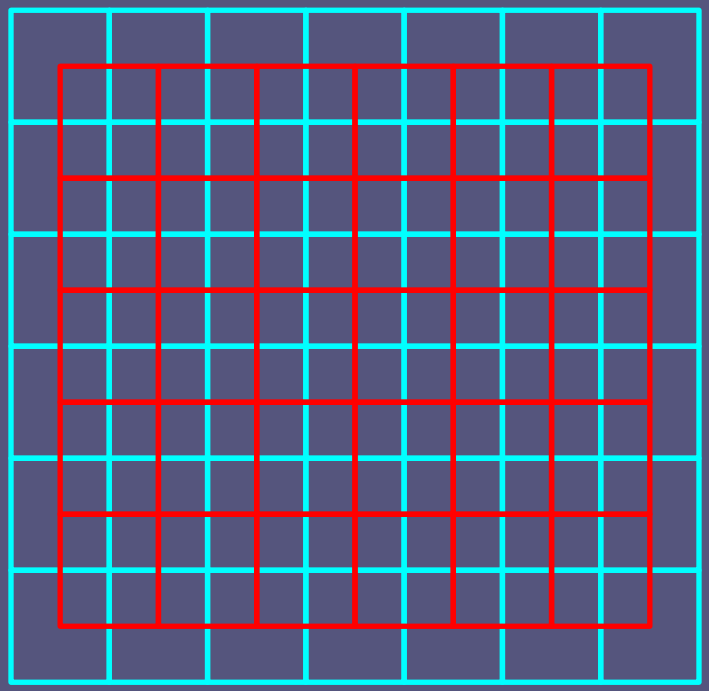


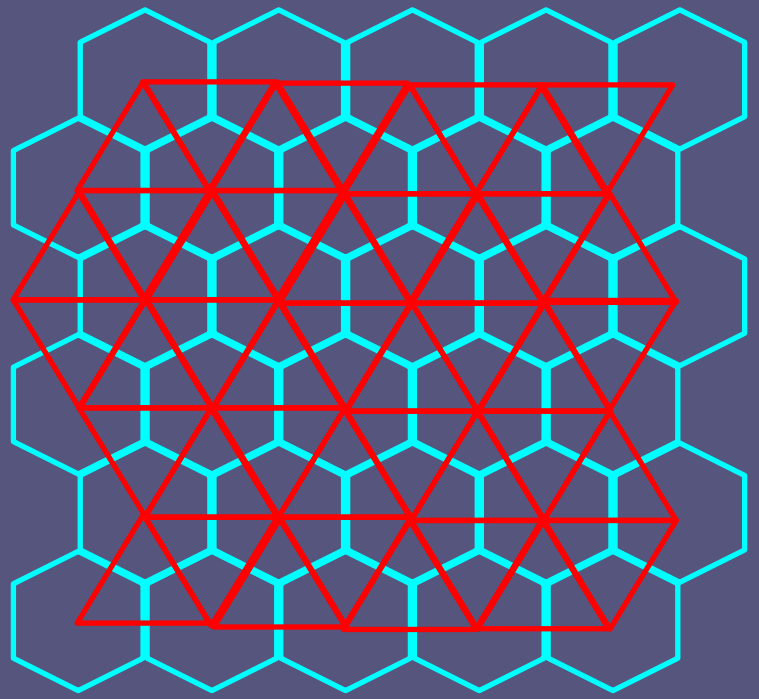
Dualita mřížek

- Duální mřížka
  - Z ploch udělám uzly
  - Z původních uzlů se stanou nové plochy
  - Hrany se „otočí o 90°“
    (velmi obrazně řečeno)

Dualita mřížek

- Pravoúhlá ⇔ Pravoúhlá
- Trojúhelníková ⇔ Šestiúhelníková

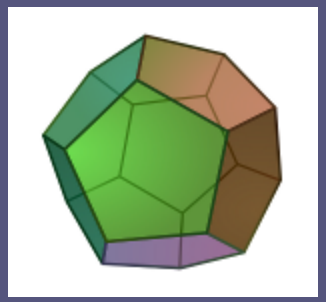


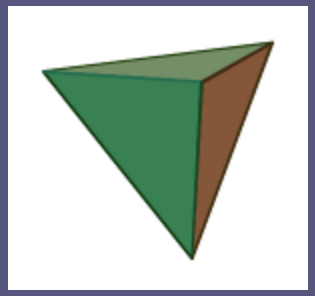


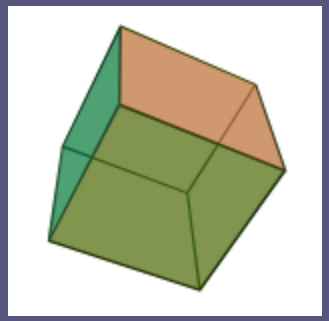
Vícerozměrné mřížky

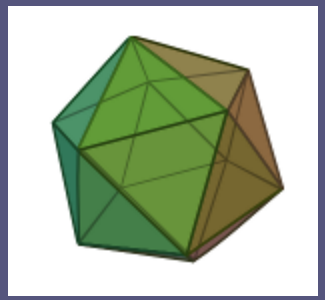
- 3-D
  - Trojrozměrné pole
- n-D
  - Vlastní mapovací funkce
- Povrch tělesa
  - Pravidelné n-stěny
  - Kopací míč

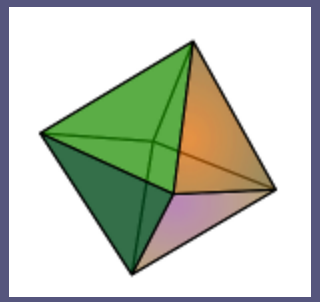
Pravidelná tělesa







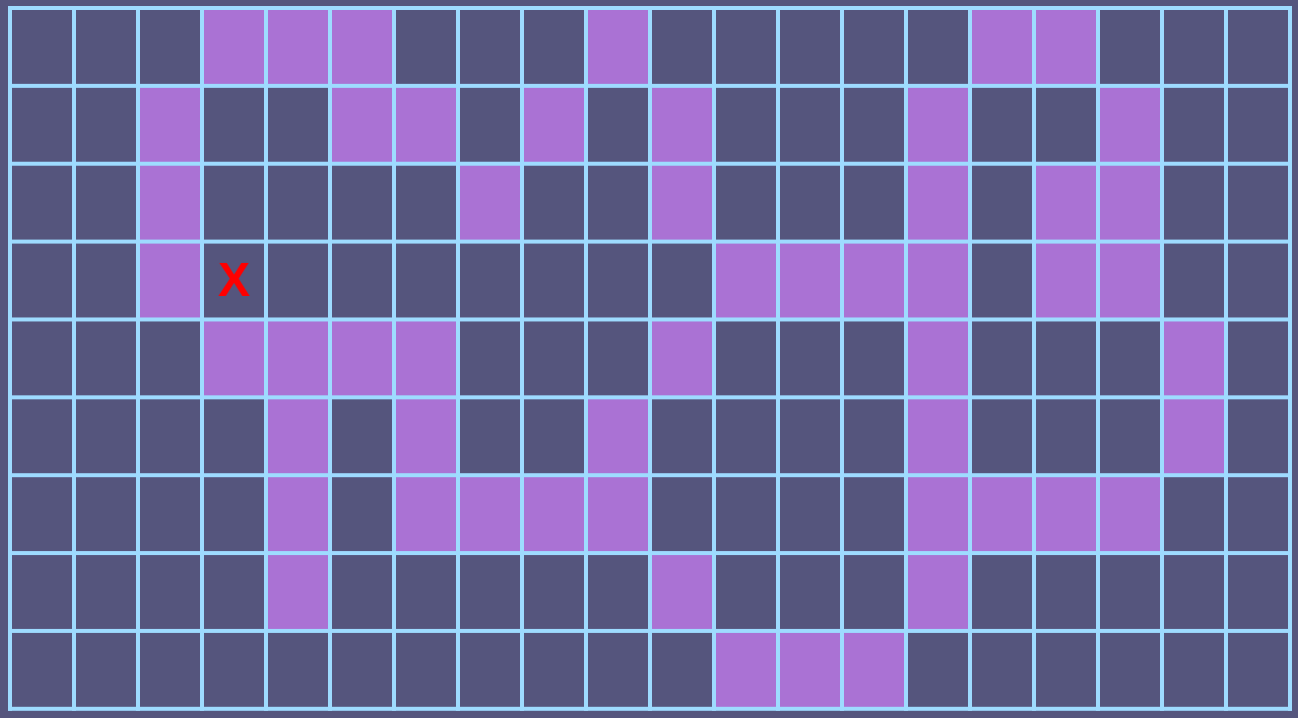




Povrch fotbalového míče

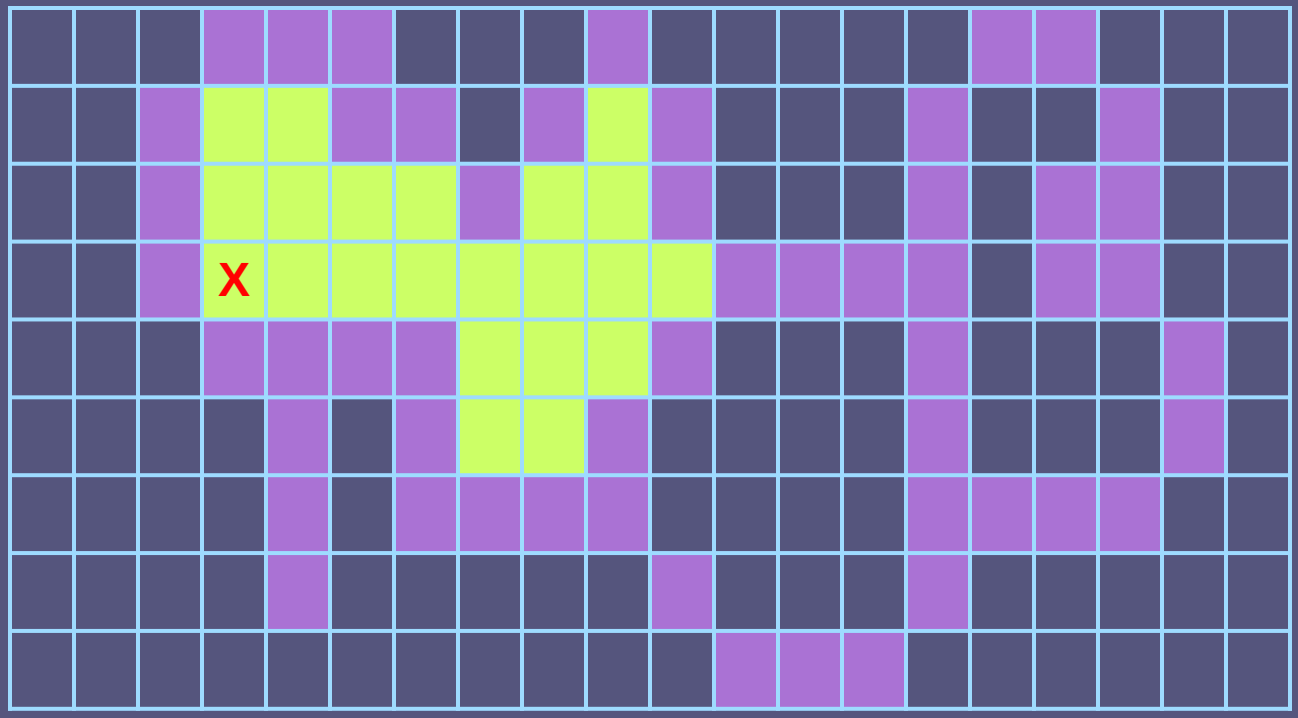
Flood Fill (Seed Fill)

- „Semínkové“ / „záplavové“ vyplňování
- Souvislá oblast **mřížky** od zadaného pole



Flood Fill (Seed Fill)

- „Semínkové“ / „záplavové“ vyplňování
- Souvislá oblast **mřížky** od zadaného pole



Flood Fill

- Princip algoritmu
  1. Vyplníme zadané políčko
  2. Přesuneme se na jeho volné sousedy

- Sousedů může být více než jeden
  - Musíme postupně
    (jedním začít a pamatovat si ostatní)

Flood Fill – sousední políčka

„Pamatovat si ostatní“ a vrátit se k nim

- Postupně pomocí rekurze
- Uložení do datové struktury
  - Fronta
  - Zásobník

Flood Fill – kostra algoritmu

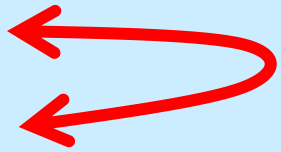
(psáno částečně pseudokódem)

```
void flood_fill(int x, int y) {
  if (!volné_pole(x,y)) return;
  zaplň_pole(x,y);
  for (sx,sy: sousední_pole(x,y)) {
     flood_fill(sx, sy);
  }
}
```

Flood Fill – jednoduchý případ

(nyní již reálný kód)

```
void flood_fill(int x, int y) {
  if (full[x][y]) return;
  full[x][y] = 1;
  flood_fill(x, y-1);
  flood_fill(x, y+1);
  flood_fill(x-1, y);
  flood_fill(x+1, y);
}
```



Pořadí!!

Flood Fill – drobné rozšíření

- Spočítat, kolik políček vyplním

```
int flood_fill(int x, int y) {
  if (full[x][y]) return 0;
  full[x][y] = 1;
  return 1 + flood_fill(x, y-1)
     + flood_fill(x, y+1)
     + flood_fill(x-1, y)
     + flood_fill(x+1, y);
}
```

Flood Fill – s frontou

- Pozor na vhodnou velikost fronty (pole)
  - Maximálně počet všech políček v mřížce

```
int[] qx, qy;
int qb, qe;
void enqueue(int x, int y) {
  if (!full[x][y]) {
     full[x][y] = 1;
     qx[qe] = x; qy[qe++] = y;
} }
```

Flood Fill – s frontou

```
void flood_fill(int x, int y) {
  qb = qe = 0;
  enqueue(x, y);
  while (qb < qe) {
     x = qx[qb]; y = qy[qb++];
     enqueue(x, y-1);
     enqueue(x, y+1);
     enqueue(x-1, y);
     enqueue(x+1, y);
} }
```

Flood Fill – pořadí vyplňování

- Rekurze nebo zásobník (DFS)
  - Preference dle „pořadí“ sousedů
  - Relativně nevyzpytatelné
- Fronta (BFS)
  - Dle vzdálenosti od počátku

Flood Fill – se vzdáleností

```
void flood_fill(int x, int y) {
  qb = qe = 0;
  enqueue(x, y, 0);
  while (qb < qe) {
     x = qx[qb]; y = qy[qb++];
     enqueue(x, y-1, dist[x][y]);
     enqueue(x, y+1, dist[x][y]);
     enqueue(x-1, y, dist[x][y]);
     enqueue(x+1, y, dist[x][y]);
} }
```

Flood Fill – se vzdáleností

```
int[] qx, qy;
int qb, qe;

void enqueue(int x, int y, int d) {
  if (!dist[x][y]) {
     dist[x][y] = d+1;
     qx[qe] = x; qy[qe++] = y;
  }
}
```

Vzdálenost – příklad

- Fialová pole jsou nedostupná („zeď“)

	8	7	6	5	4		12	13	
					3		11	12	
	10		2	**1**	2		10		
	9		3				9	10	
	8		4	5	6	7	8	9	
	7	6	5	6	7	8	9	10	

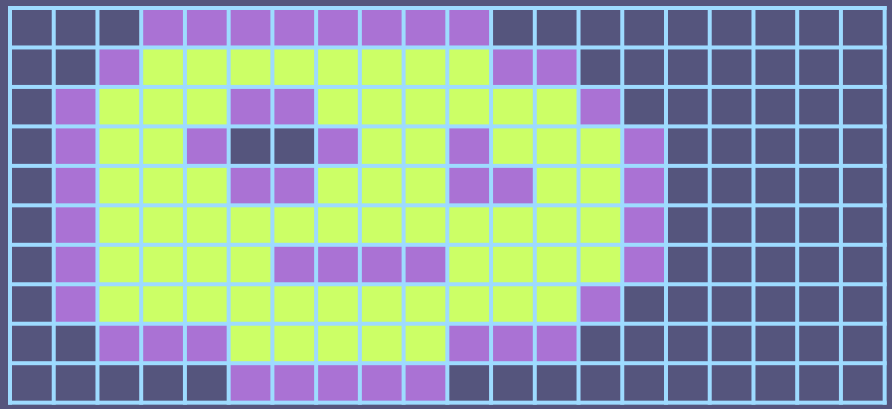
Příklady z úvodu

- Maximální prázdný/plný obdélník
  - Dynamické programování (viz EP1)

- Detekce souvislých oblastí
- Spočítání oblastí („skvrn“)
- Vyplnění oblasti
- Průchod bludištěm
- *… a další*

Příklady z úvodu

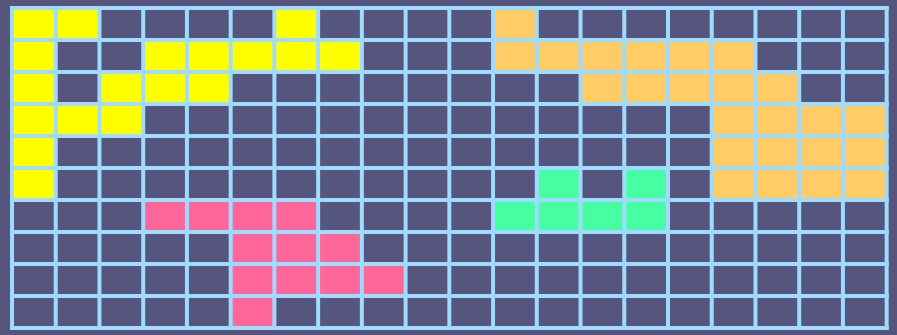
- Vyplnění oblasti
- → Flood-Fill
  - BFS nebo DFS (jen pozor na zásobník)



Příklady z úvodu

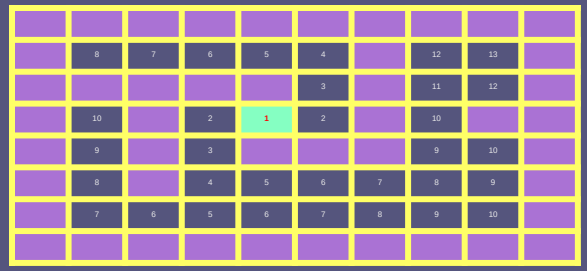
- Detekce souvislých oblastí
- → Flood-Fill
  - BFS nebo DFS (jen pozor na zásobník)

- Spočítání oblastí („skvrn“)
  - Projdu všechna políčka, pro obsazená zavolám Flood-Fill („vymazání“)



Příklady z úvodu

- Průchod bludištěm
- → BFS
  - Najdu rovnou nejkratší cestu



Úlohy

- Mřížky
  - **Hexagonal Routes**
  - **Triangular Vertices**
- Záplavové vyplňování
  - **Jeskyně**
  - **Die Is Cast**

Tvorba úloh

Vaše „autorská úloha“

- Zadání
- Vzorové řešení
- Testovací data
- Rozbor časového limitu

Autorská úloha – zadání

- Čeština / angličtina (slovenština)
  - Přesné
  - Jednoznačné
  - Stručné
  - Formát vstupu a výstupu
  - Jednoduché příklady
- *Každý by měl správně pochopit…*

Autorská úloha – řešení

- C / C++ / Java / Python
- Správné
- Přehledné
- S vysvětlením algoritmu (česky/anglicky)
  - Na začátku zdrojového kódu
  - Ve zvláštním souboru

- *Řeší všechny případy & vysvětlí postup…*

Autorská úloha – testovací data

- Odpovídající zadání (formát i data)
- Postihující maximum případů
  - Nejmenší a největší případy
  - Speciální případy
- Mohou být generována programem
  - Několik ručních „ukázkových“ vstupů

- *Nesprávný program by neměl projít…*

Autorská úloha – časový limit

- Stačí krátké zdůvodnění
  - Jak moc je časový limit důležitý
  - Násobek času vzorového programu
  - Na čem selže neefektivní řešení (je-li takové)
- Ne (!) konkrétní časy

- *Návod pro toho, kdo by ho chtěl nastavit…*

Autorská úloha – odevzdání

- Termín **28. dubna 2026**
  - (cca **2** týdny před zápočtem)
- Hodnocení kolegů
  - Bez uvedení autora
  - Snažíte se opravdu hodnotit úlohu

Autorská úloha – hodnocení

- Podmínka zápočtu
- Celkem max. **15 bodů**
  - 4 body – „estetický dojem“
    - Vhodnost tématu
    - Správná obtížnost
    - Nápaditost, příběh
  - 7 bodů – faktická správnost
    - Ovlivněna objevenými chybami
  - 4 body – vzájemné hodnocení