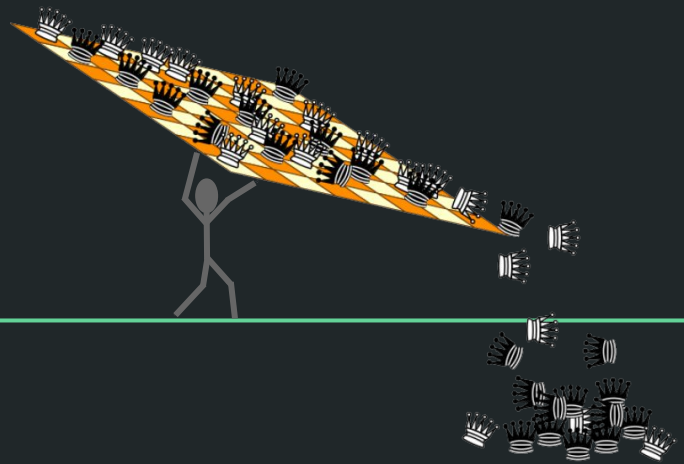
# PilsProg 2021

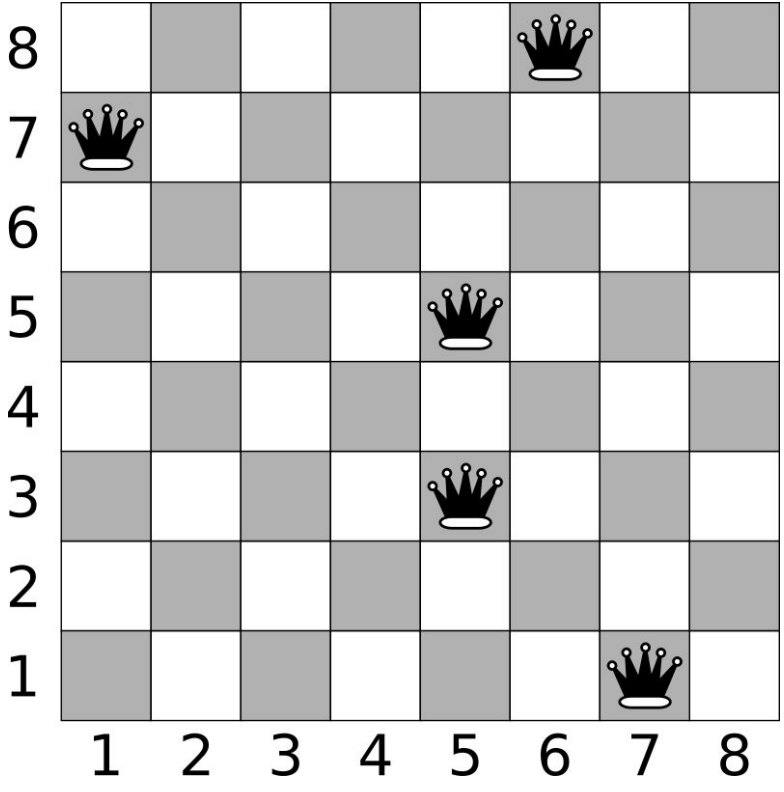
Řešení finálových úloh

Martin Maňák
27. 3. 2021

# Dámy v ohrožení



# Dámy v ohrožení 1/2



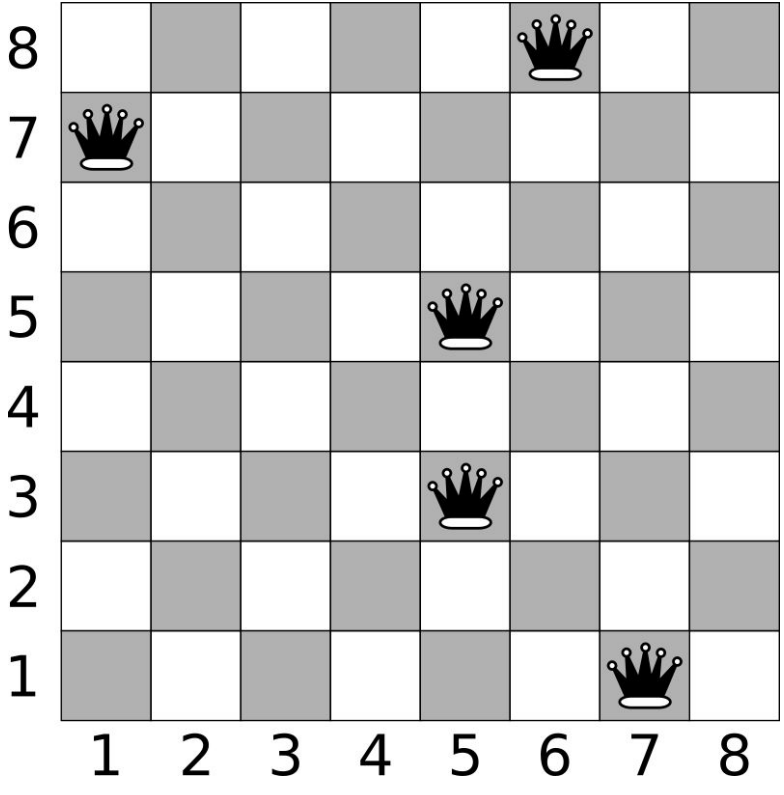

1. Každou dámu započítat do jejího řádku, sloupce, diagonál
2. Pro každou dámu spočítat ostatní dámy ve stejném řádku, sloupci a na diagonálách
3. Vrátit maximum

```
n = velikost šachovnice
int row[n], col[n], diag1[2n], diag2[2n]
```

Bod 1:

```
for q in queens:
    (r, c) = (q.row - 1, q.col - 1)
    row[r]++
    col[c]++
    diag1[r + c]++
    diag2[n - 1 - c + r]++
```

# Dámy v ohrožení 2/2



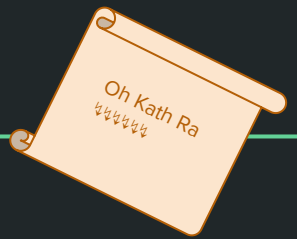

1. Každou dámu započítat do jejího řádku, sloupce, diagonál
2. Pro každou dámu spočítat ostatní dámy ve stejném řádku, sloupci a na diagonálách
3. Vrátit maximum

```
n = velikost šachovnice
int row[n], col[n], diag1[2n], diag2[2n]
```

Body 2 a 3:

```
m = 0
for q in queens:
    (r, c) = (q.row - 1, q.col - 1)
    m = max(m, row[r] + col[c]
        + diag1[r + c]
        + diag2[n - 1 - c + r] - 4)
return m
```

# Šifrovaná zpráva



# Šifrovaná zpráva

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| T | E | N | I | S |
| S | L | O | Z | I |
| T | O | S | T | J |
| E | L | I | N | E |
| A | R | N | I | _ |

TENIS
SLOZITOST JE LINEARNI

| 1 | 3 | 2 | 4 | 0 |
|---|---|---|---|---|
| E | I | N | S | T |
| L | Z | O | I | S |
| O | T | S | J | T |
| L | N | I | E | E |
| R | I | N | _ | A |

TENIS
LOLRZTNIOSINIJE_STEA

## Zakódování zprávy

1. Z textu odstranit mezery (tyto nebudou kódovány)
2. Vytvořit tabulku (počet sloupců podle klíče)
3. Zapsat text po řádcích do tabulky
4. Poslední řádek tabulky doplnit mezerami
5. Uspořádat sloupce tabulky podle písmena v záhlaví
6. Projít tabulku po sloupcích, ze znaků vytvořit řetězec

## Dekódování zprávy

1. Vytvořit tabulku (počet sloupců podle klíče)
2. Zapsat text po sloupcích do tabulky
3. P = permutace seřazení znaků klíče
   $P^{-1}$ = permutace inverzní
4. Změnit pořadí sloupců tabulky podle $P^{-1}$
5. Projít tabulku po řádcích, ze znaků vytvořit řetězec
6. Odstranit mezery na konci řetězce

# Ostrovy

# Ostrovy 1/14

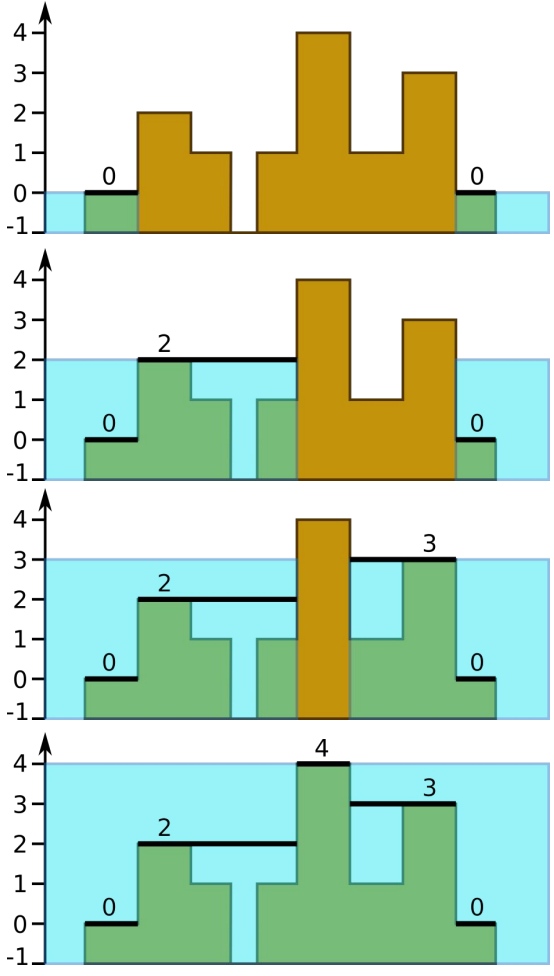
- **Připravit záplavovou mapu**

  ZM[i][j] = výška hladiny, na které bude políčko (i, j) poprvé zaplaveno vodou

  Výpočet pomocí simulace **vzestupu** hladiny z 0 do maxima pomocí BFS s prioritní frontou, která servíruje níže položená políčka dřív než výše položená políčka.

  BFS začíná na vnějším okraji mapy (hladina 0). Do aktuálně zpracovávaného políčka ukládá maximum z jeho výšky a maxima výšek dříve zpracovaných políček.

  Sousednost: 4-okolí ← ↑ → ↓



# Ostrovy 2/14

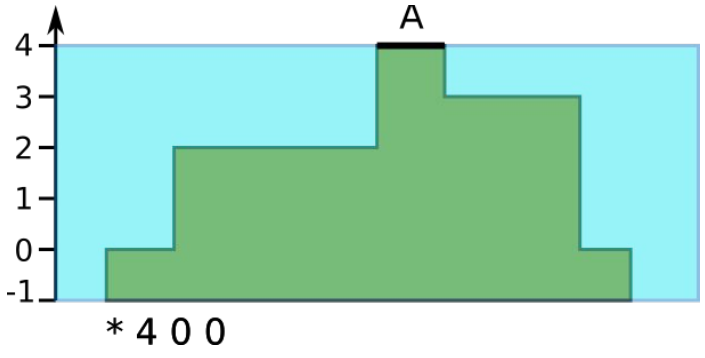
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 3/14

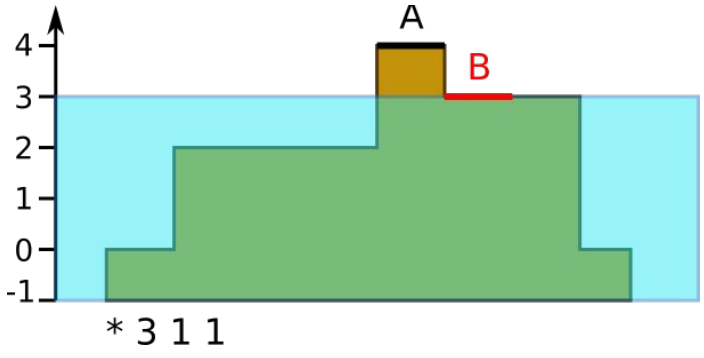
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 4/14

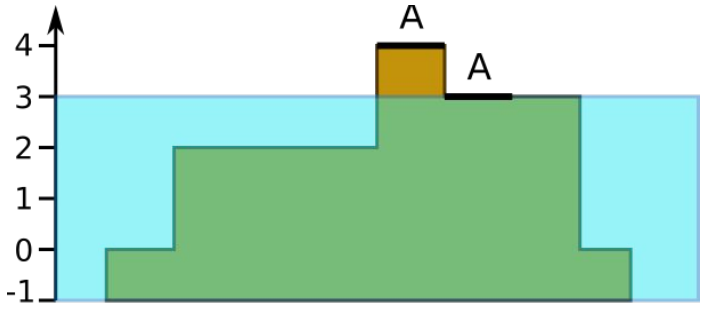
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 5/14

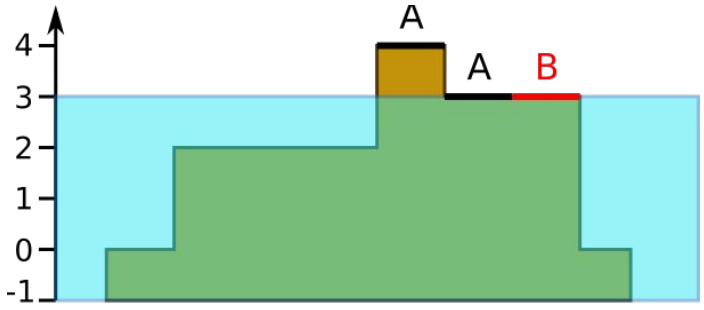
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 6/14

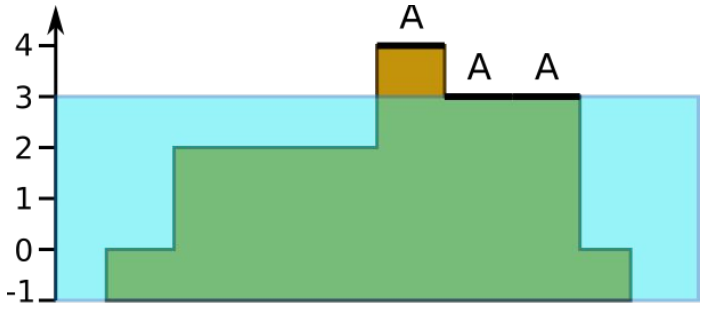
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 7/14

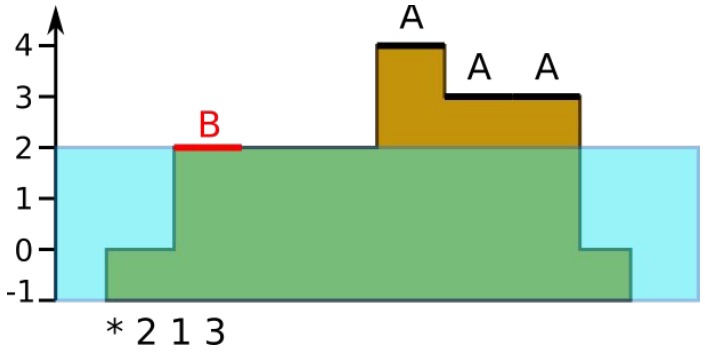
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 8/14

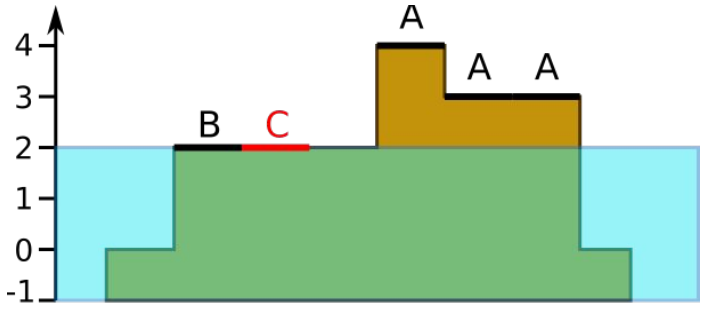
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 9/14

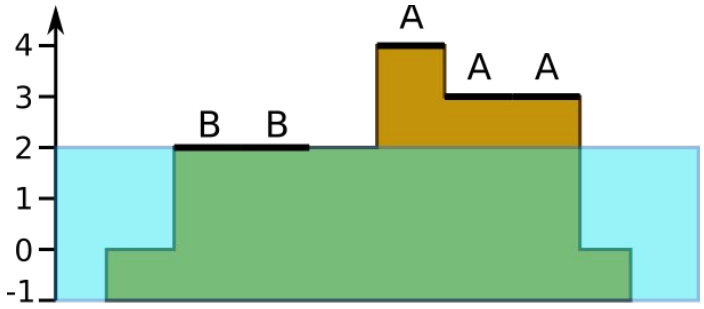
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 10/14

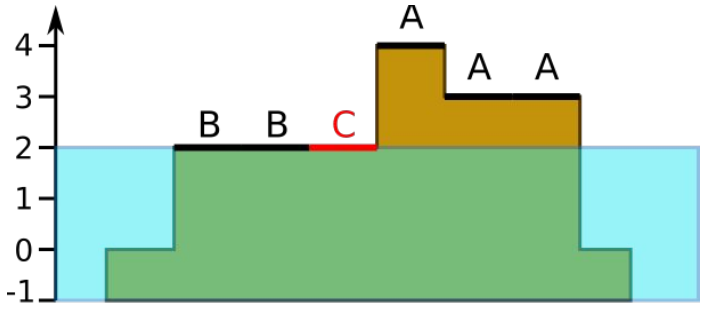
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 11/14

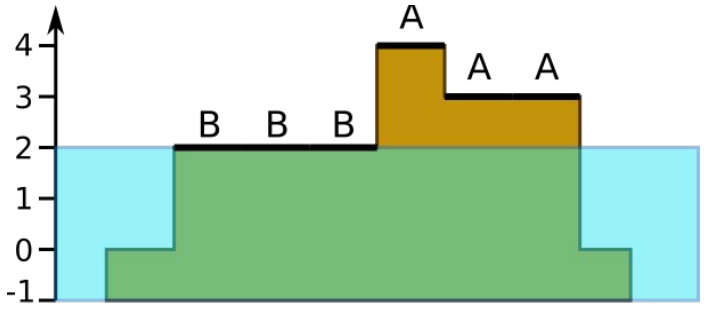
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 12/14

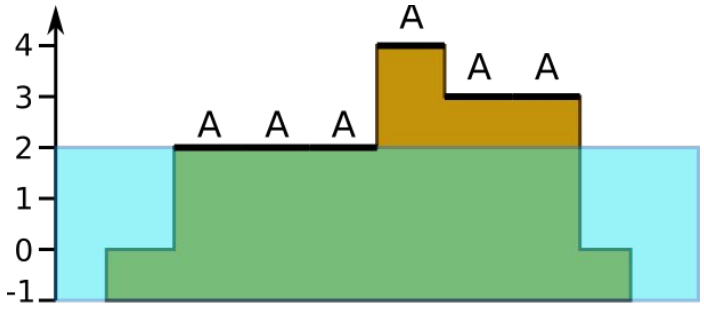
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.



# Ostrovy 13/14

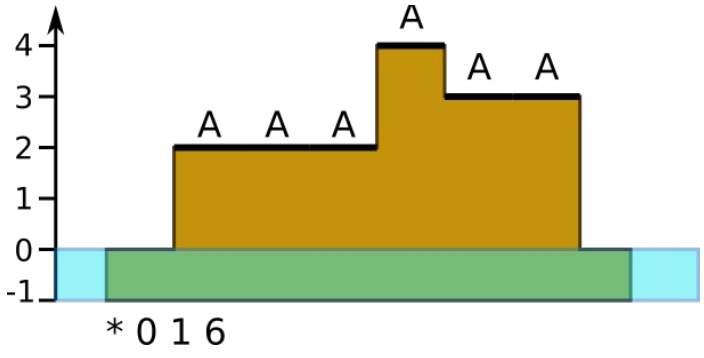
- Na ZM simulovat **sestup hladiny vody** z maxima až do 0:

  Seznam políček (i, j) seřadit sestupně podle ZM[i][j] a zpracovat v tomto pořadí ⇒ **objevování políček ostrovů**

  Každé objevené políčko připojit do skupiny políček existujících ostrovů (až 8 sousedů). Použít disjoint-set DS (union-find) pro udržování a detekci spojování skupin.

  U každé skupiny udržovat počet členů (rozloha ostrova). Spojení skupin ⇒ úprava rozlohy ⇒ úprava mediánu.

  Těsně před zpracováním každého nového políčka: Hlídat změnu hladiny, počtu ostrovů, mediánu rozlohy oproti poslednímu výsledku, a buď nahradit poslední výsledek nebo přidat další do seznamu. Na konec seznam otočit.

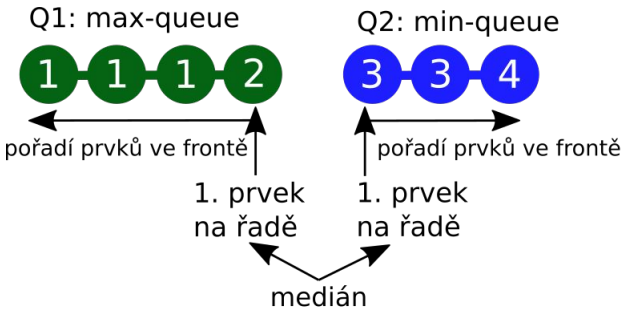


# Ostrovy 14/14

- **Dynamicky udržovaný medián**:

  a) Dynamicky udržovaná seřazená posloupnost prvků, strom (AVL nebo red-black), nalezení prostředního prvku nebo dvojice prvků

  b) Dvojice prioritních front (max- a min-queue), vyvažování “přesýpáním” prvků z jedné do druhé

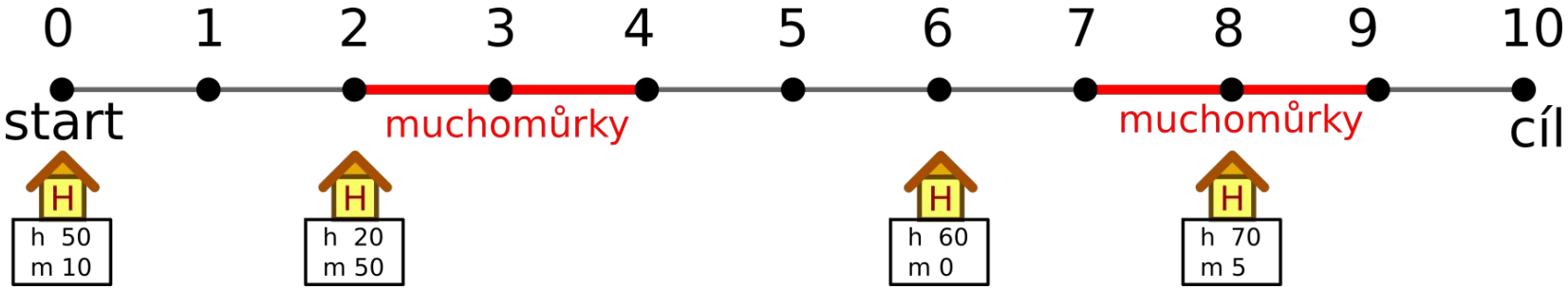



Vyváženost front: 0 <= |Q1| - |Q2| <= 1

Přidání prvku:
např. nejdřív vložím do Q1, pak vyjmu 1. prvek, který je na řadě v Q1 a dám ho do Q2.

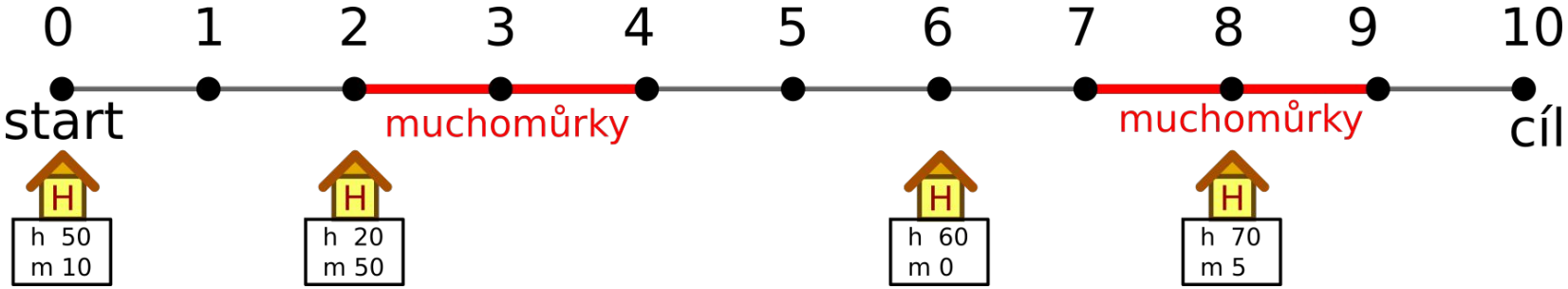
# Zachraňte Sněhurku

# Zachraňte Sněhurku 1/2




- Úkol: Spočítat nejlevnější cenu přesunu ze startu do cíle
- Spřežení veverek žere 1 houbu / km (muchomůrka nebo hříbek)
- Omezená nosnost - max. 100 hub
- Vyznačené úseky - nutné krmit veverky pouze muchomůrkami
- V některých místech lze koupit houby (ne nutně oba druhy, různé ceny)

# Zachraňte Sněhurku 2/2




- M[i, j, k] = minimální cena ze startu do i-tého kilometru, aby stav zásob byl j hříbků a k muchomůrek: min(a, b, c, d)

  (a) M[i - 1, j + 1, k] konzumace 1 hříbku ($i > 0$, $j + 1 + k \leq$ nosnost a i není “muchomůrkový”)

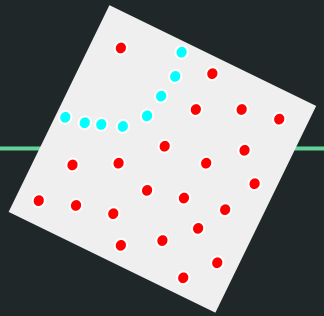
  (b) M[i - 1, j, k + 1] konzumace 1 muchomůrky (pokud $i > 0$ a $j + k + 1 \leq$ nosnost)

  (c) M[i, j - 1, k] + h nákup 1 hříbku na pozici i za cenu h (pokud $j > 0$ a $h \neq 0$)

  (d) M[i, j, k - 1] + m nákup 1 muchomůrky na pozici i za cenu m (pokud $k > 0$ a $m \neq 0$)

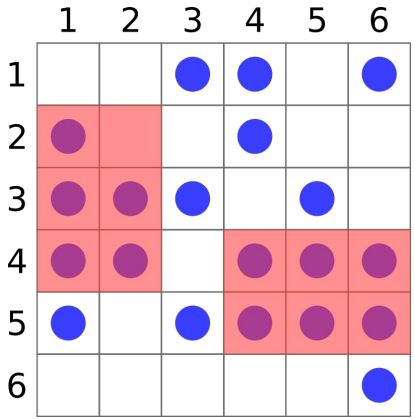
- Výsledek: min( M[cíl, *, *] )

# Moderní umění



# Moderní umění 1/5

- Máme zadaný rastr n x n políček, některá políčka obsahují tzv. *tečky*
- Úkol: Najít všechny obdélníky obsahující alespoň m teček a zabírající minimální možný počet políček

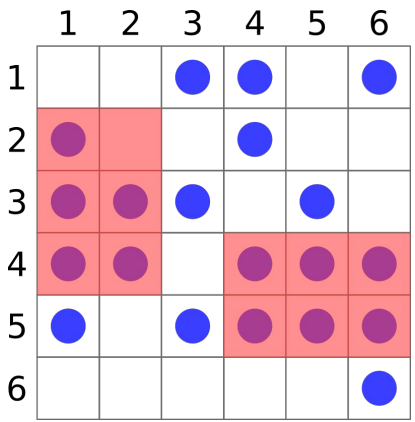



2 z 10 výřezů o minimální ploše obsahující alespoň 5 teček

# Moderní umění 2/5

- Prefixový součet ve 2D
  PS[i, j] = počet teček v obdélníku od (1,1) po (i,j)

- Počet teček v libovolném výřezu pomocí PS v konstantním čase
  pt(i, j, k, l) = počet teček od (i, j) po (k, l) včetně




2 z 10 výřezů o minimální ploše obsahující alespoň 5 teček

PS[i, j]

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 2 | 3 |
| 1 | 1 | 2 | 4 | 4 | 5 |
| 2 | 3 | 5 | 7 | 8 | 9 |
| 3 | 5 | 7 | 10 | 12 | 14 |
| 4 | 6 | 9 | 13 | 16 | 19 |
| 4 | 6 | 9 | 13 | 16 | 20 |

12 = 10 + 8 - 7 + **1**

pt(i, j, k, l)

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 2 | 3 |
| 1 | 1 | 2 | 4 | 4 | 5 |
| 2 | 3 | 5 | 7 | 8 | 9 |
| 3 | 5 | 7 | 10 | 12 | 14 |
| 4 | 6 | 9 | 13 | 16 | 19 |
| 4 | 6 | 9 | 13 | 16 | 20 |

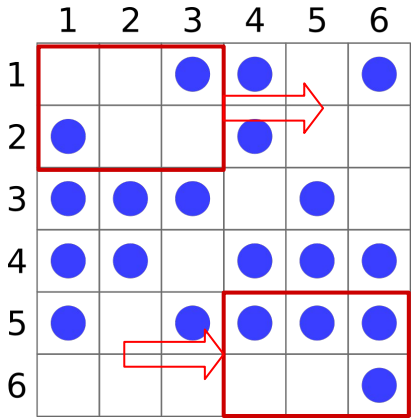
6 = 19 - 9 - 9 + 5

# Moderní umění 3/5

- Pro okno šířky w a výšky h dokážeme spočítat

  max_pt(w, h) = maximální počet teček zachytitelný oknem w x h

  Posun okna w x h z levého horního rohu do pravého dolního rohu. Počítání teček pomocí funkce pt(i, j, i + h - 1, j + w - 1). Nalezení maxima.




2 z 10 výřezů o minimální ploše obsahující alespoň 5 teček

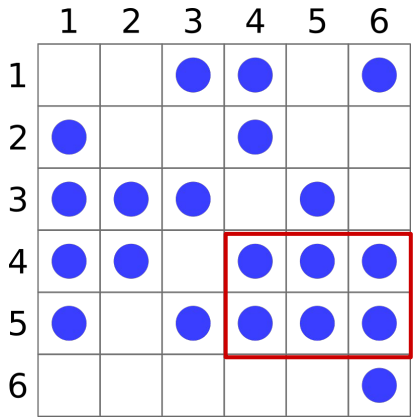
# Moderní umění 4/5

- Okno rozměru w x h je tzv. *přijatelné*, je-li alespoň 1x1 a max_pt(w, h) ≥ m
- Uvažujme *matici přijatelnosti* indexovanou velikostí okna

  MP[h, w] = 1 pokud je okno w x h přijatelné, jinak 0
- Je-li w x h přijatelné, pak jsou přijatelné i všechna širší a všechna vyšší okna
- V MP mezi 0 a 1 vzniká linie jedniček, kterou lze vytrasovat v čase O(n) a vyhnout se tak vyhodnocování všech políček

w

| h \ MP | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| 3 | 0 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 1 | 1 | 1 |



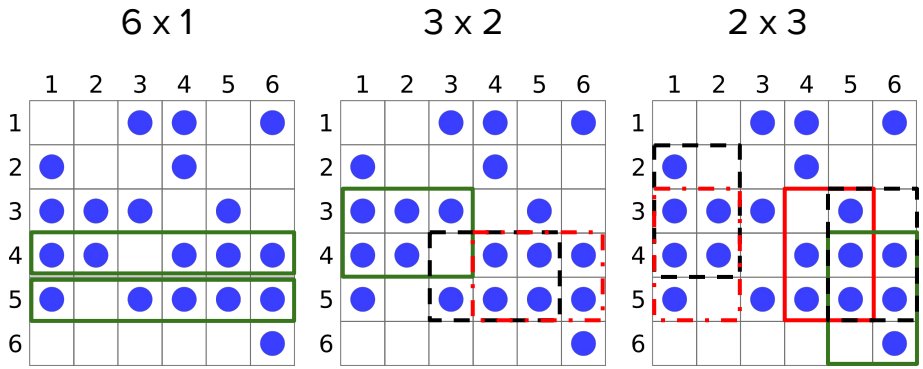
# Moderní umění 5/5

- V MP mezi 0 a 1 vzniká linie jedniček
- Z této linie vybereme přijatelné rozměry oken s minimální plochou
- Pro každý takový rozměr projdeme celou matici, najdeme všechna konkrétní vyhovující okna a seřadíme je podle zadání

w

h

| MP | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| 3 | 0 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 1 | 1 | 1 |



# Děkuji za pozornost

Úlohy připravili

Martin Maňák
Zuzana Majdišová
Tomáš Potužák

Kontakt: manak at kiv.zcu.cz