# Rekurze

IB111 Programování a algoritmizace

2010

# Rekurze

- použití funkce při její vlastní definici
- volání sebe sama (s jinými parametry)

# Faktoriál

$$n! = 1 \cdot 2 \cdots (n-1) \cdot n$$

$$f(n) = \begin{cases} 1 & \text{if } n = 1 \\ n \cdot f(n-1) & \text{if } n > 1 \end{cases}$$

# Faktoriál iterativně (pomocí cyklu)

```
def fact(n):
    f = 1
    for i in range(1,n+1):
        f = f * i
    return f
```

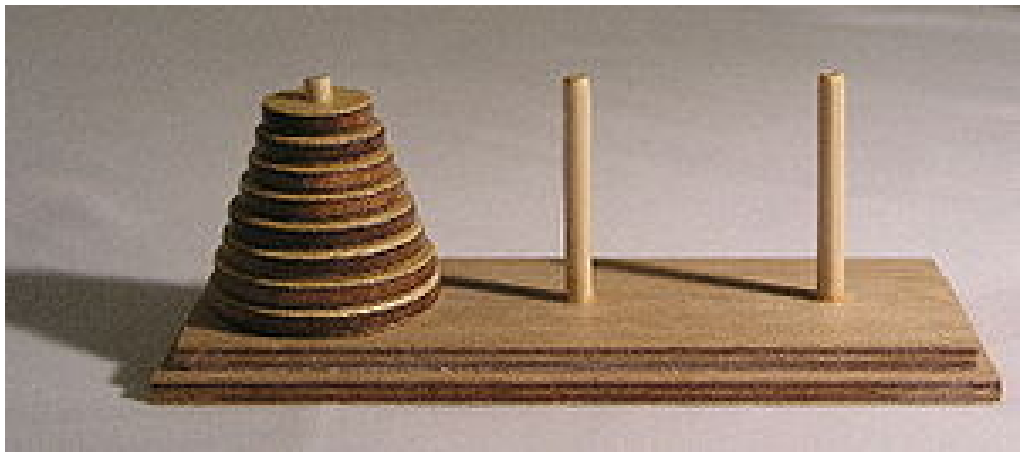
# Faktoriál rekurzivně

```
def fact(n):
    if n == 1: return 1
    else: return n * fact(n-1)
```

# Hanojské věže aneb O konci světa

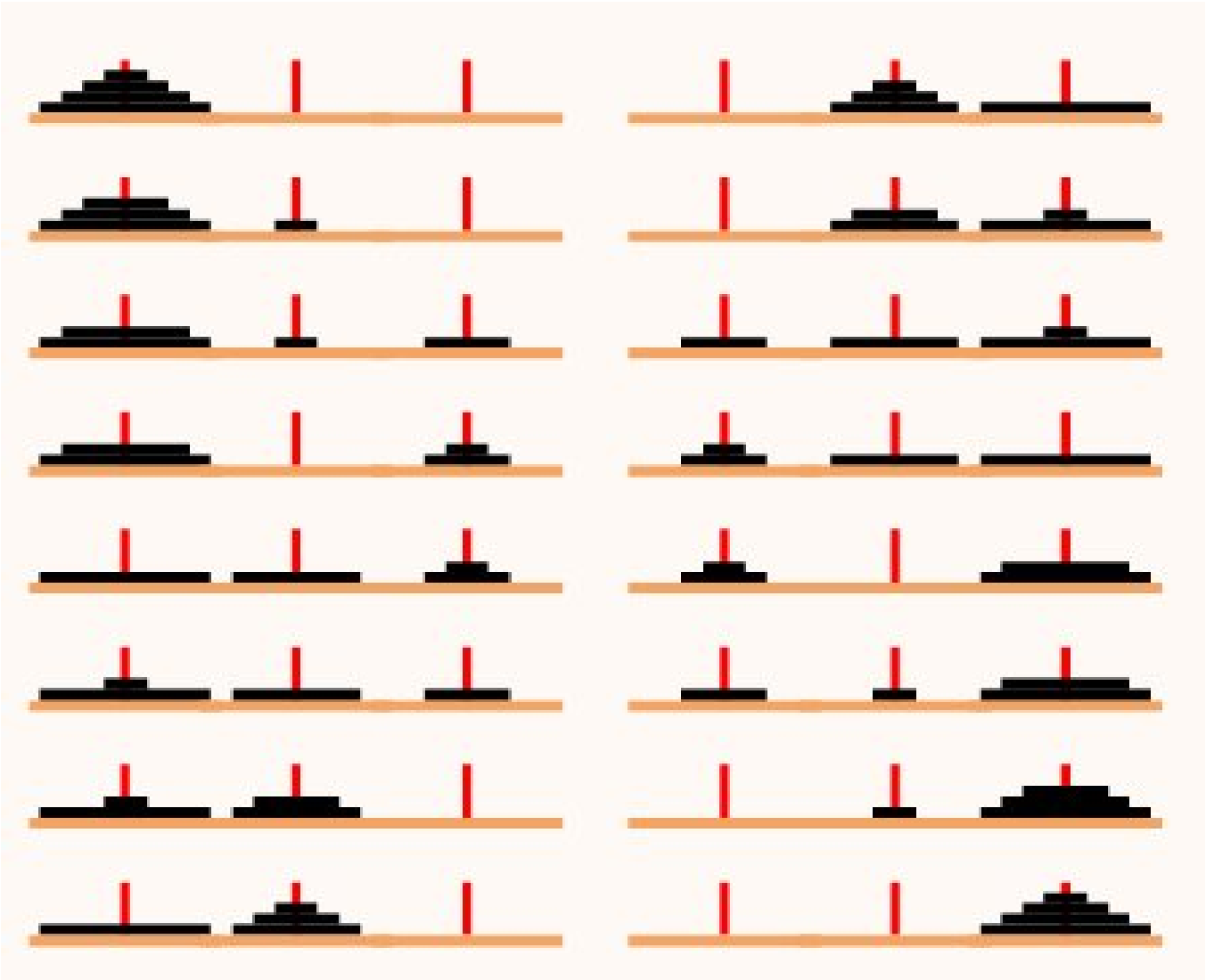
- video:
  `http://www.fi.muni.cz/~xpelanek/IB111/hanojske_veze/`
- klášter kdesi vysoko v horách u města Hanoj
- velká místnost se třemi vyznačenými místy
- 64 různě velkých zlatých disků
- podle věštby mají mniši přesouvat disky z prvního na třetí místo
- a až to dokončí ...

# Hanojské věže: pravidla

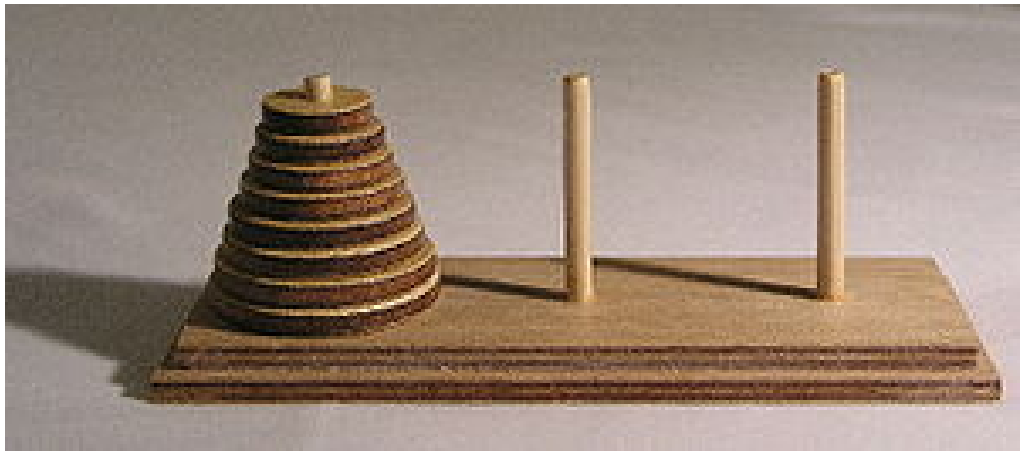


- $N$ disků různých velikostí naskládaných na sobě
- vždy může být jen menší disk položen na větším
- možnost přesunout jeden horní disk na jiný kolíček
- cíl: přesunout vše z prvního na třetí

# Hanojské věže: řešení



# Hanojské věže: rekurzivní řešení



```
přesun(N, odkud, kam, kudy)
  pokud (N=1) táhni: odkud -> kam
  jinak
    přesun(N-1, odkud, kudy, kam)
    přesun(1, odkud, kam, kudy)
    přesun(N-1, kudy, kam, odkud)
```

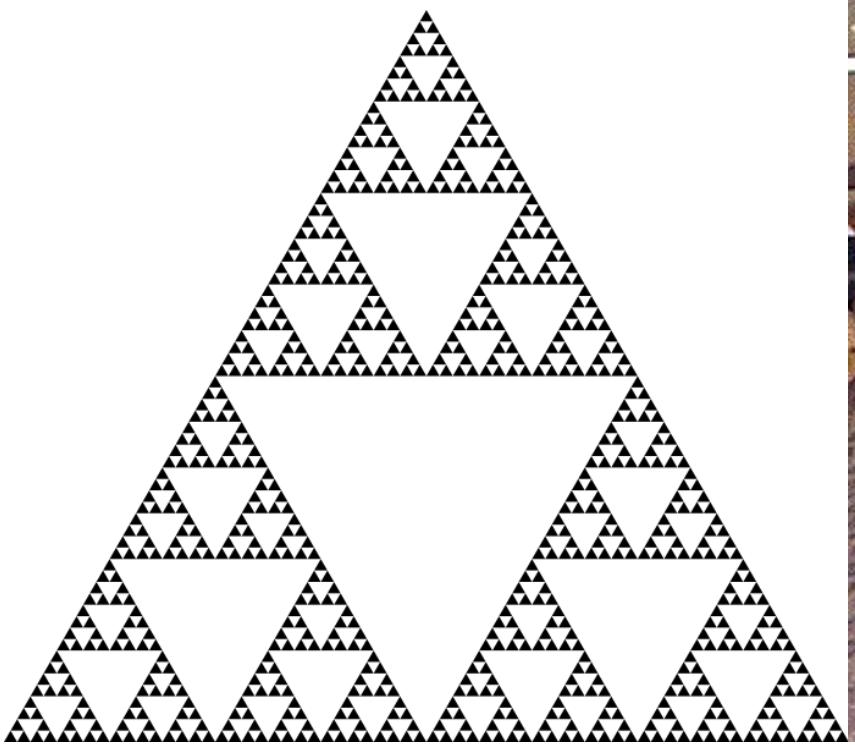
# Odbočka: nečekané souvislosti

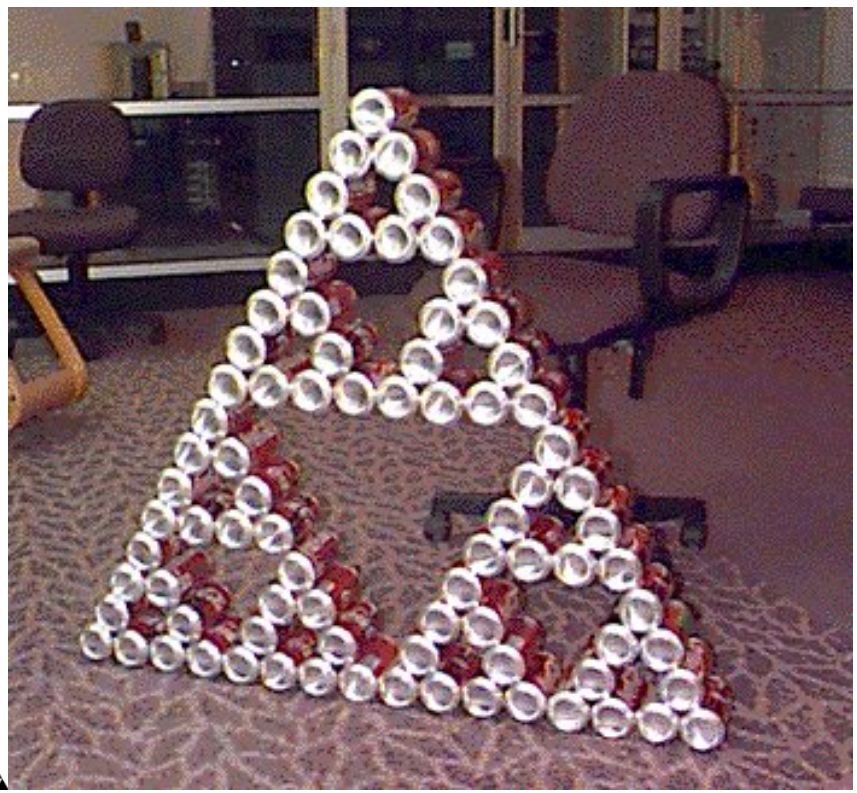
- Hanojské věže
- fraktál: Sierpinského košík
- Paskalův trojúhelník

# Sierpinského košík

rekurzivně definovaný geometrický útvar

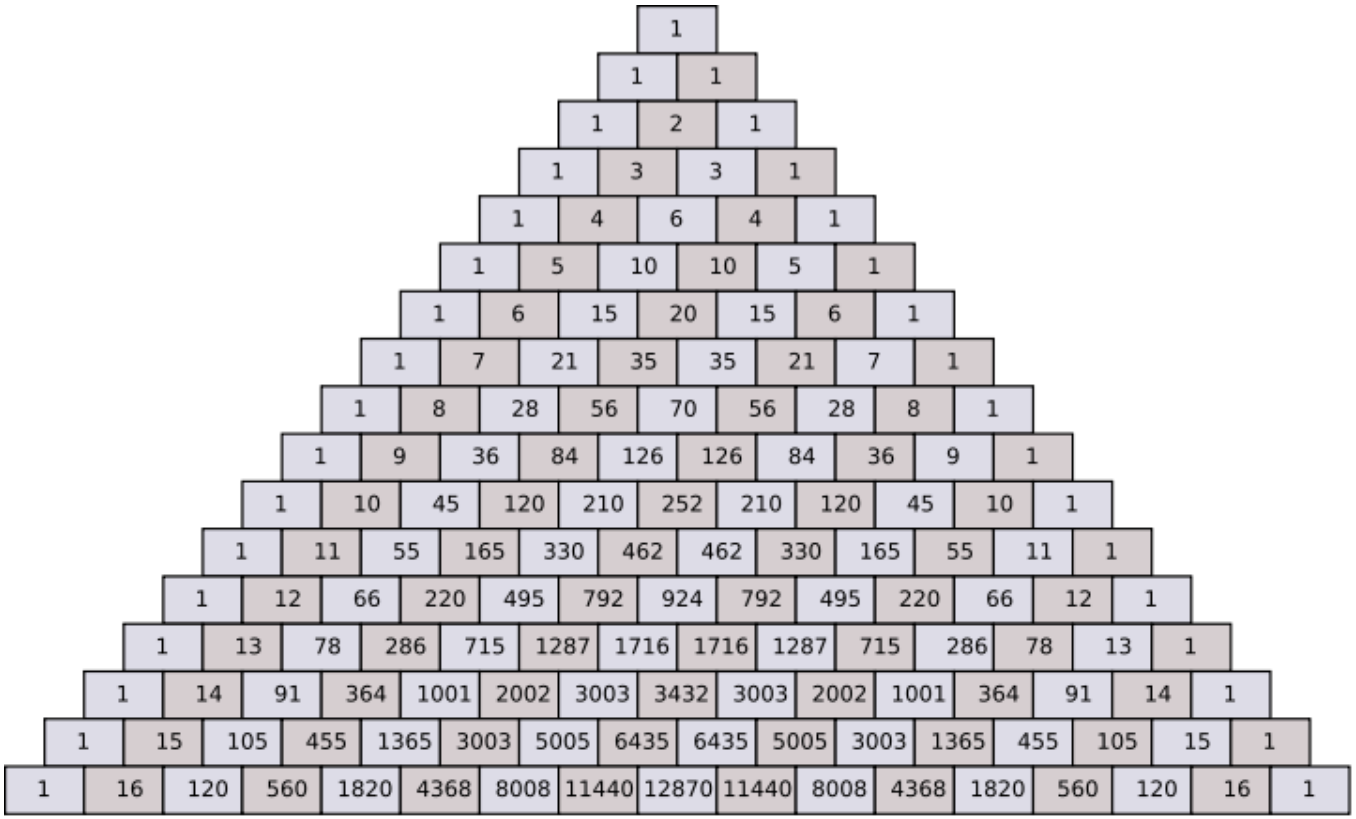
# Sierpinského košík



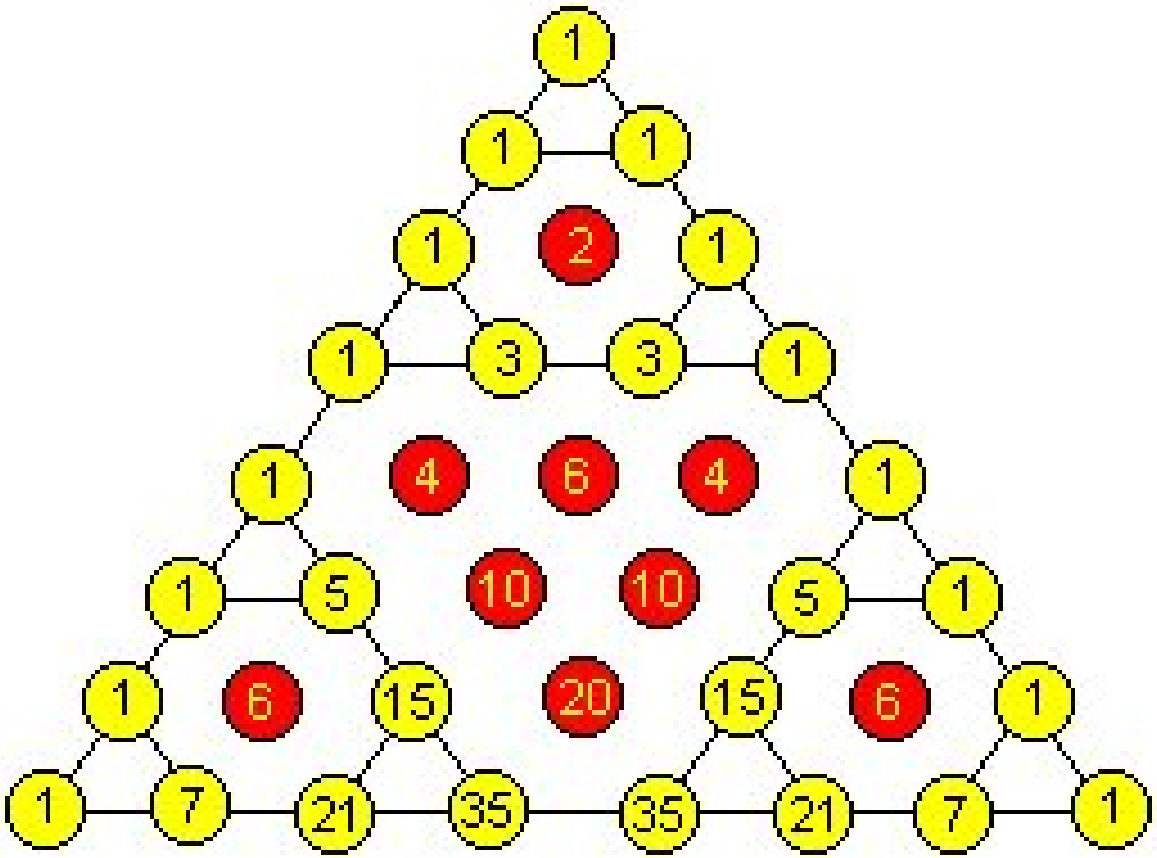


# Paskalův trojúhelník

```
binom(n,k) = binom(n-1,k-1) + binom(n-1,k)
```



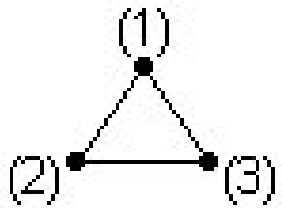
# Paskalův trojúhelník a Sierpinského košík

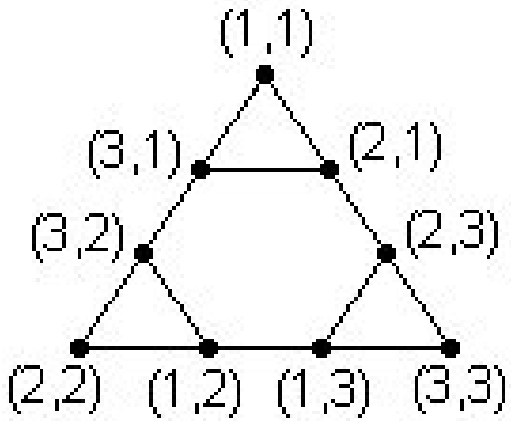


# Co to má společného s Hanojskými věžmi?

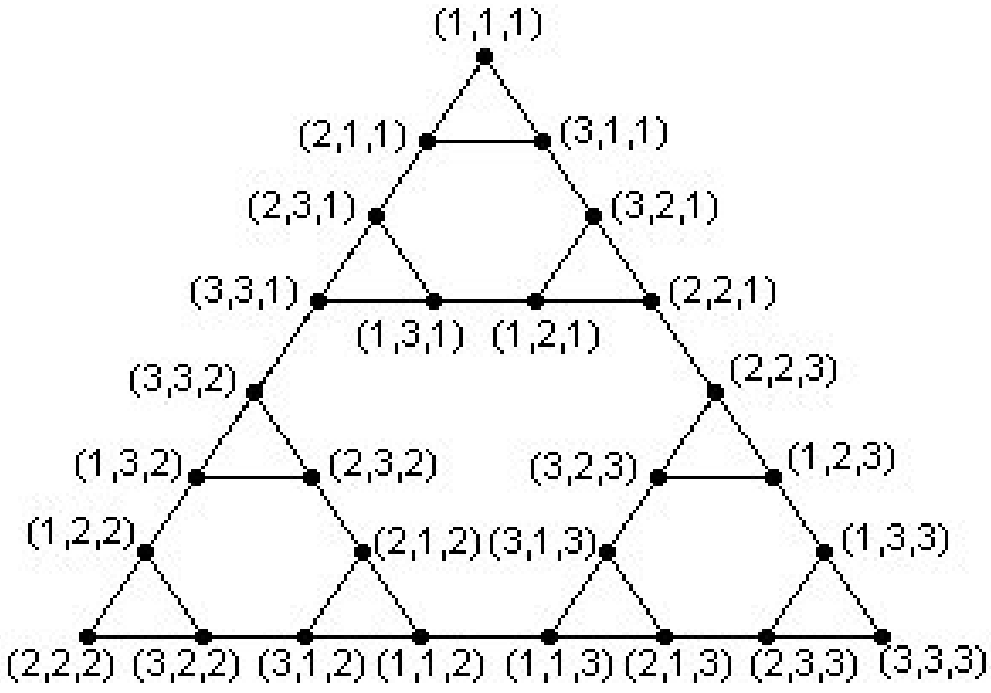
- jak vypadá stavový prostor úlohy?
- připomenutí, stavový prostor:
  - graf (uzly + hrany)
  - uzly = konfigurace hry
  - hrany = povolené tahy
- jak vypadá stavový prostor pro 1 disk? pro 2 disky?

# Hanojské věže: stavový prostor



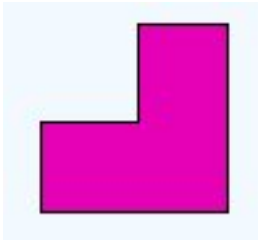
# Hanojské věže: stavový prostor



# Hanojské věže: variace

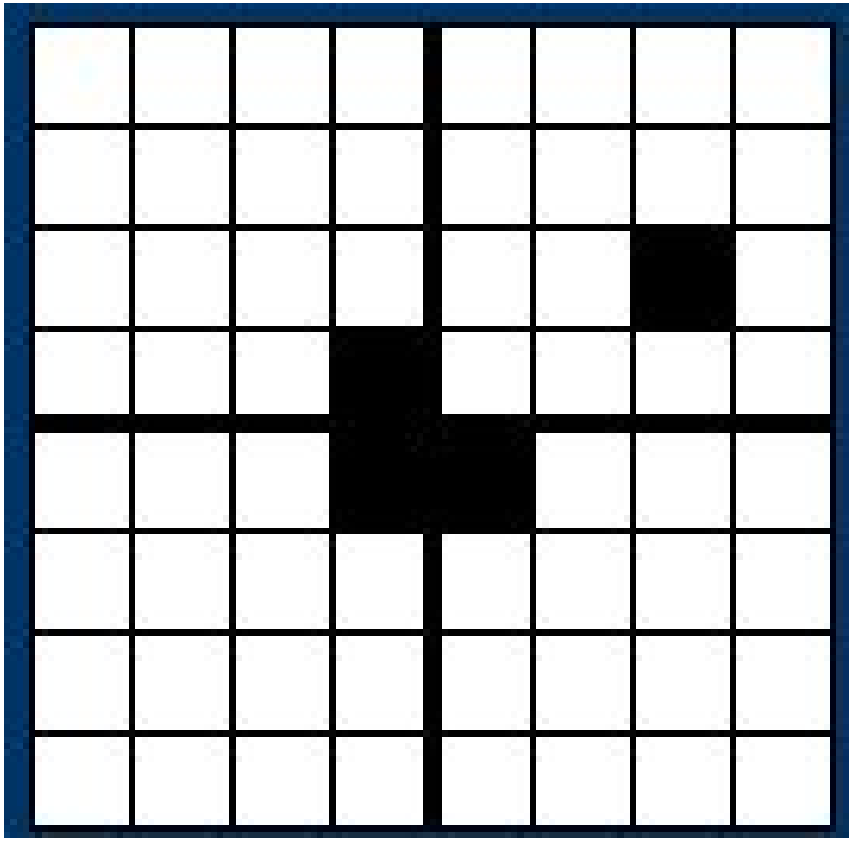
- přesun z obecné konfigurace do obecné konfigurace
- více než 3 kolíky

# Pokrývání plochy L kostičkami



- mřížka 8x8 s chybějícím levým horním polem
- úkol: pokrýt zbývající políčka pomocí L kostiček
- rozšíření:
  - chybějící libovolné pole
  - obarvení 3 barvami, aby sousedi byli různí

# Řešení



- rozdělit na čtvrtiny
- umístit jednu kostku
- rekurzivně aplikovat řešení na jednotlivé části

# Aplikace rekurze

- Euclidův algoritmus – NSD
- prohledávání grafu do hloubky
- vyhledávání opakovaným půlením
- třídění (quicksort, mergesort)
- generování permutací, kombinací
- fraktály

# Vyhledávání opakovaným půlením

- hra na 20 otázek
- hledání v intervalu
- hledání v binárním stromu

# Vyhledávání půlením intervalu

**Algorithm 6.2** BINARYSEARCHREC
**Input:** An array $A[1..n]$ of $n$ elements sorted in nondecreasing order and an element $x$.
**Output:** $j$ if $x = A[j], 1 \leq j \leq n$, and 0 otherwise.

1. *binarysearch*$(1, n)$

**Procedure** *binarysearch*$(low, high)$

1. **if** $low > high$ **then return** 0
2. **else**
3. $\quad mid \leftarrow \lfloor (low + high)/2 \rfloor$
4. $\quad$ **if** $x = A[mid]$ **then return** $mid$
5. $\quad$ **else if** $x < A[mid]$ **then return** *binarysearch*$(low, mid - 1)$
6. $\quad$ **else return** *binarysearch*$(mid + 1, high)$
7. **end if**

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

# Třídění

- quicksort
  - vyber pivota
  - rozděl na menší a větší
  - zavolej quicksort na podčásti
- mergesort
  - rozděl na polovinu
  - každou polovinu setřid pomocí mergesort
  - spoj obě poloviny

# Generování permutací, kombinací

- permutace množiny = všechna možná pořadí
  - příklad: permutace množiny $\{1, 2, 3, 4\}$
  - jak je vypsat systematicky?
  - jak využít rekurzi?
- $k$-prvkové kombinace $n$-prvkové množiny = všechny možné výběry $k$ prvků
  - příklad: 3-prvkové kombinace množiny $\{A, B, C, D, E\}$
  - jak je vypsat systematicky?
  - jak využít rekurzi?

# Nevhodné použití rekurze

- ne každé použití rekurze je efektivní
- Fibonačiho posloupnost (králíci):

$$f_1 = 1$$

$$f_2 = 1$$

$$f_n = f_{n-1} + f_{n-2}$$

# Shrnutí

- rekurze: využití rekurze pro definici sebe sama
- příklady, hádanky: Hanojské věže, L kostičky
- aplikace: DFS, vyhledávání