

Rekurze

IB111 Programování a algoritmizace

2011

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

Faktoriál

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

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

Faktoriál iterativně (pomocí cyklu)

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

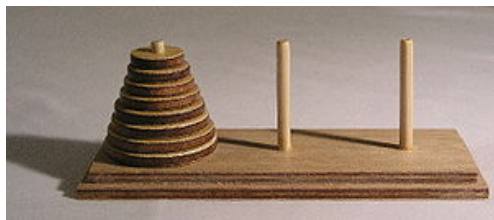
Faktoriál rekurzivně

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

Hanojské věže aneb O konci světa

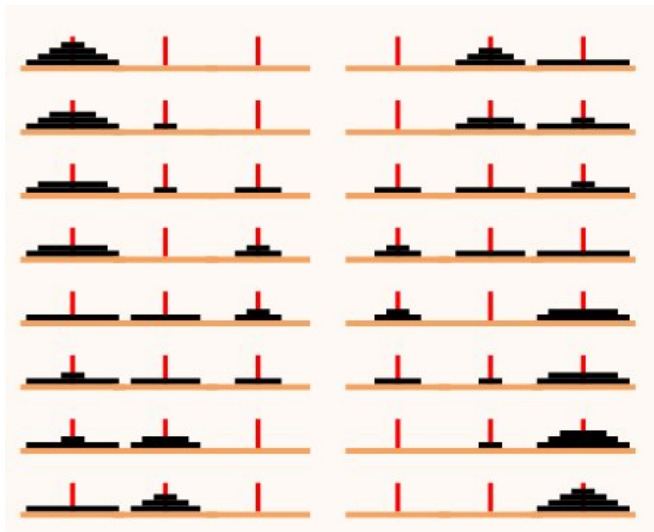
- video:
http://www.fi.muni.cz/~xpelane/IB111/hanojske_veze/
- klášter kdesi vysoko v horách u města Hanoj
- velká místnost se třemi vyznačenými místy
- 64 různě velikých zlatých disků
- podle věštby mají mniši přesouvat disky z prvního na třetí místo
- a až to dokončí ...

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

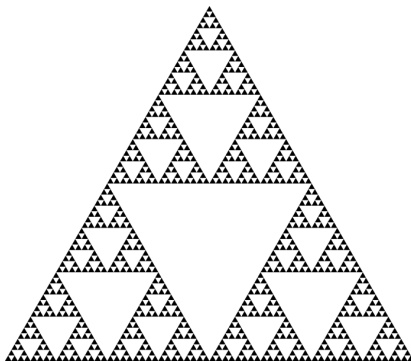
- Hanojské věže
- fraktál: Sierpinského košík
- Paskalův trojúhelník

Sierpinského košík

rekurzivně definovaný geometrický útvar



Sierpinského košík

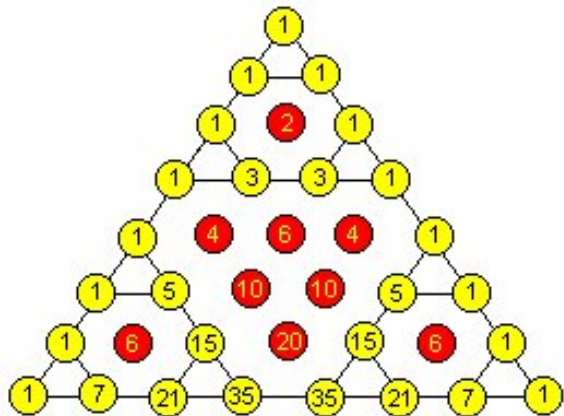


Paskalův trojúhelník

$$\text{binom}(n,k) = \text{binom}(n-1,k-1) + \text{binom}(n-1,k)$$

1																
1	1															
1	2	1														
1	3	3	1													
1	4	6	4	1												
1	5	10	10	5	1											
1	6	15	20	15	6	1										
1	7	21	35	35	21	7	1									
1	8	28	56	70	56	28	8	1								
1	9	36	84	126	126	84	36	9	1							
1	10	45	120	210	252	210	120	45	10	1						
1	11	55	165	330	462	462	330	165	55	11	1					
1	12	66	220	495	792	924	792	495	220	66	12	1				
1	13	78	286	715	1287	1716	1716	1287	715	286	78	13	1			
1	14	91	364	1001	2002	3003	3432	3003	2002	1001	364	91	14	1		
1	15	105	455	1365	3003	5005	6435	6435	5005	3003	1365	455	105	15	1	
1	16	120	560	1820	4368	8008	11440	12870	11440	8008	4368	1820	560	120	16	1

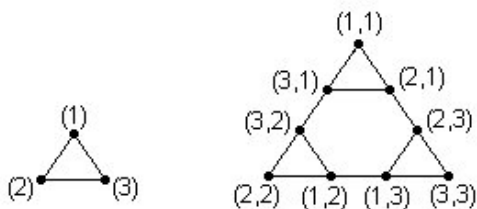
Pascalův trojúhelník a Sierpinského košík



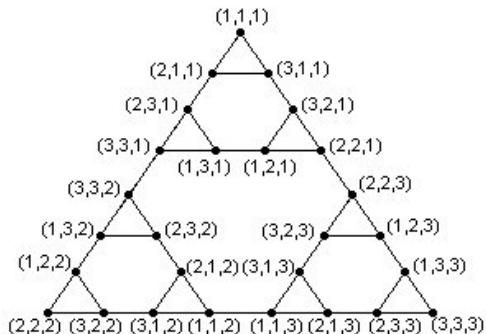
Co to má společného s Hanojskými věžmi?

- 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



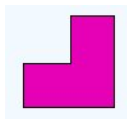
Hanojské věže: stavový prostor



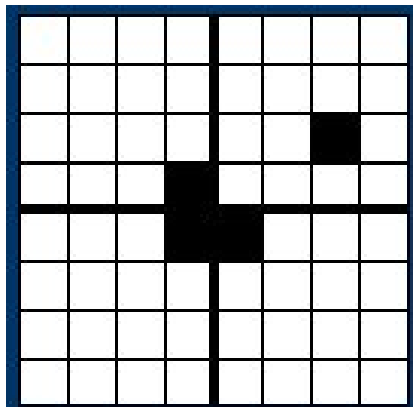
Hanojské věže: variace

- 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



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

Aplikace rekurze

- Euclidův algoritmus – NSD
- prohledávání grafu do hloubky
- vyhledávání opakovaným půlením
- řadící algoritmy (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. $\text{binarysearch}(1, n)$

Procedure $\text{binarysearch}(low, high)$

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

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

Řadící algoritmy

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

Generování permutací, kombinací

- permutace množiny = všechna možná pořadí
 - příklad: permutace množiny $\{1, 2, 3, 4\}$
 - jak je vypsát 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 vypsát systematicky?
 - jak využít rekurzi?

Nevhodné použití rekurze

- ne každé použití rekurze je efektivní
- Fibonacciho 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