Příklad 1/17

Vypočtěte, kolik celkem času zabere jedno zavolání funkce **rekur(5);** za předpokladu, že provedení přikazu **xyz();** trvá vždy jednu milisekundu a že dobu trvání všech ostatních akcí zanedbáme.

```
void rekur(int x) {
   if (x < 1) return;
   rekur(x-1);
   xyz();
   rekur(x-2);
 }
```

Příklad 2/17

Určete, jakou hodnotu vypíše program po vykonání příkazu **print(rekur(4));**, když rekurzivní funkce **rekur()** je definována takto:

```
int rekur(int x) {
  if (x < 1) return 2;
  return (rekur(x-1)+rekur(x-1));
}
```

Napište vztah, který vyjadřuje velikost vrácené hodnoty v závislosti na vstupní hodnotě x. Zároveň určete, pro které hodnoty x bude vztah definován s uvážením rozsahu typu `int`.

Příklad 3/17

Charakterizujte slovy, jakou hodnotu vrátí funkce ff v závislosti na hodnotách jejích vstupních parametrů.

Nepopisujte kód samotný, pouze návratovou hodnotu.

```
int ff(int x, int y) {
  if (x > 0) return ff(x-1,y)+y;
  return 0;
}
```

Příklad 4/17

Předpokládejte, že na vstupu funkce ff jsou dvě celá čísla, hodnota xx je kladná, hodnota y je nezáporná.

Doplňte na vyznačená místa kód tak, aby funkce vracela hodnotu x^y.

```
int ff(int x, int y) {
  if (.....) return .........;
  return ........;
 }
```

Příklad 5/17

Daná funkce ff je volána s parametry a, b: **ff(a, b);**, přičemž a i b jsou celá kladná čísla. Napište vztah, který určí, kolikrát bude volána funkce **abc(x)**, v závislosti na hodnotě parametrů a, b.

```
void ff(int x, int p) {
  if (x > 0) ff(x-p);
  abc(x);
  if (x > 0) ff(x-p);
}
```

Příklad 6/17

Napište rekurzivní funkci, která pro zadané číslo N vypíše řetězec skládající se z N jedniček následovaných 2N dvojkami. Pro dané N bude funkce volat sama sebe právě N krát.

Např. pro N = 2 vypíše 111222222.

Příklad 7/17

Pomocí rekurzivní funkce vypište pro zadané kladné číslo N posloupnost čísel

1 2 ... N−2 N−1 N N N−1 N−2 ... 2 1

Příklad 8/17

Určete, kolik *znaků* vypíše na výstup funkce rec zavolaná s parametrem 20:

```
void rec(int val) {
   if (val < 1) return;
   rec(val-1);
   printf("%d%s",val," ");
   rec(val-1);
 }
```

Příklad 9/17

Rekurzivní algoritmus A dělí úlohu o velikosti *n* na 2 stejné části, pro zisk výsledku musí každou tuto část zpracovat dvakrát. Čas potřebný na rozdělení úlohy na části a na spojení dílčích řešení je úměrný hodnotě *n*. Asymptotická složitost algoritmu A je popsána rekurentním vztahem

a) $T(n) = 4T(n/2) + n$

b) $T(n) = n \cdot T(n \cdot 4/2)$

c) $T(n) = T(n/2) + 4n/2$

d) $T(n) = 2T(n/4) + n$

e) $T(n) = n \cdot T(n/2) + n \cdot \log(n)$

Příklad 10/17

Rekurzivní algoritmus A dělí úlohu o velikosti n na 3 stejné části a pro zisk výsledku stačí, když zpracuje pouze dvě z nich. Čas potřebný na rozdělení úlohy na části a na spojení dílčích řešení je úměrný hodnotě n^2. Asymptotická složitost algoritmu A je popsána rekurentním vztahem

a) $T(n) = n \cdot T(n \cdot 3/2)$

b) $T(n) = T(n/3) + 2n/3$

c) $T(n) = 3T(n/2) + n^2$

d) $T(n) = n \cdot T(n/2) + n^2$

e) $T(n) = 2T(n/3) + n^2$

Příklad 11/17

Daný rekurzivní algoritmus pracuje tak, že pro $n > 1$ data rozdělí na 4 části stejné velikosti, zpracuje 5 těchto částí (tj. jednu z nich dvakrát) a pak jejich řešení spojí. Na samotné rozdělení problému a spojení řešení menších částí potřebuje dobu úměrnou hodnotě $n^2 - n$.

a) Nakreslete první tři úrovně stromu rekurze.

b) Vypočtěte hlubku stromu rekurze.

c) Metodou stromu rekurze určete asymptotickou složitost A.

d) Určete asymptotickou složitost A pomocí Mistrovské věty.

Příklad 12/17

Daný rekurzivní algoritmus A pracuje tak, že pro $n > 1$ data rozdělí na 3 části stejné velikosti, zpracuje každou tuto část dvakrát a pak jejich řešení spojí. Na samotné rozdělení problému a spojení řešení menších částí potřebuje dobu úměrnou hodnotě $\cdot\, n^{1/2} \cdot \log_2(n)$.

a) Nakreslete první tři úrovně stromu rekurze.

b) Vypočtěte hlubku stromu rekurze.

c) Metodou stromu rekurze určete asymptotickou složitost A.

d) Určete asymptotickou složitost A pomocí Mistrovské věty.

Příklad 13/17

Složitost rekurzivního algoritmu je dána rekurencí:

a) $T(n) = 3 \cdot T(\lfloor n/2 \rfloor) + n$

b) $T(n) = T(n/2) + n^2$

c) $T(n) = 4 \cdot T(n/2 + 2) + n$

Použijte metodu stromu rekurze k nalezení vhodného horního odhadu funkce $T(n)$.

Ověřte výsledek substituční metodou.

Příklad 14/17

Složitost rekurzivního algoritmu je dána rekurencí:

a) $T(n) = T(n-1) + T(n/2) + n$

b) $T(n) = T(n-a) + T(a) + c \cdot n$, kde $a \geq 1$, $c > 0$.

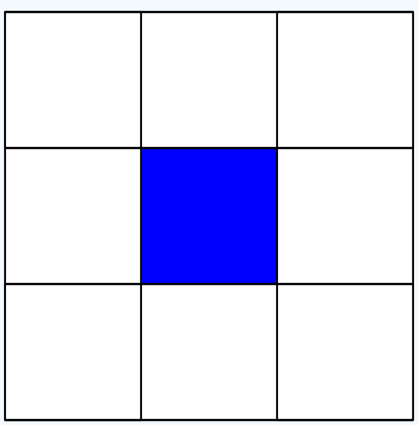
c) $T(n) = T(\alpha n) + T((1-\alpha)n) + c \cdot n$, kde $0 < \alpha < 1$, $c > 0$.

Použijte metodu stromu rekurze k nalezení vhodného horního odhadu funkce $T(n)$.

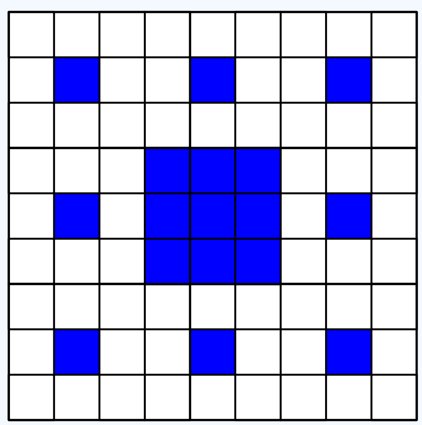
Sierpińského koberce

Prvky pole P_N o velikosti $3^N \times 3^N$ jsou pouze čísla 1 a 0. Obrázky níže znázorňují pole P_N pro několik hodnot N, modrá barva představuje hodnotu 1, bílá 0. Napište kód, který pro dané N vytvoří a vyplní pole P podle daného vzoru.

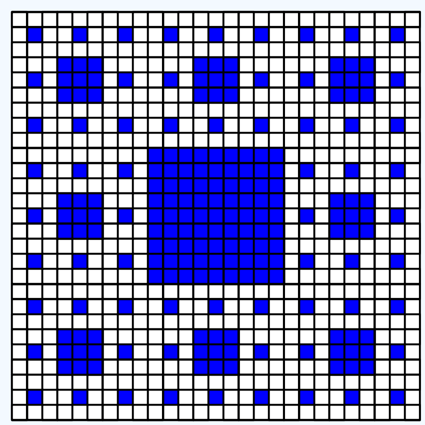
N= 1



N = 2

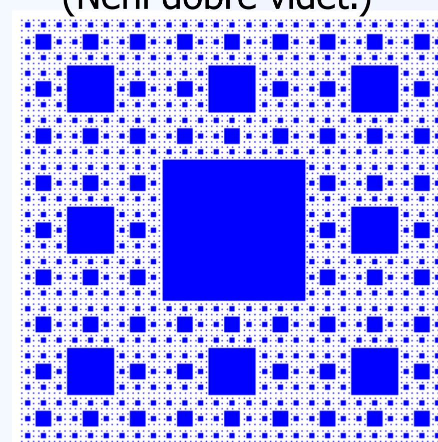


N = 3



N = 5

(Není dobře vidět.)



Příklad 16/17

Ackermannova funkce A(n, m) je definována níže.
Doplňte do tabulky na pozici [n][m] hodnotu A(n, m).

$$A(n, m) = \begin{cases} m+1 & \text{pro } n=0 \\ A(n-1,1) & \text{pro } n>0,\ m=0 \\ A(n-1,A(n,m-1)) & \text{pro } n>0,\ m>0 \end{cases}$$

***A*(*n* , *m*)**	***m* = 0**	***m* = 1**	***m* = 2**	***m* = 3**	***m* = 4**
***n* = 0**					
***n* = 1**					
***n* = 2**					
***n* = 3**					
***n* = 4**					

Příklad 17/17

Pro složitost rekurzivního algoritmu platí $T(1) = 1$.
Pro každé $n > 1$ je $T(n)$ dána rekurencí:

a) $T(n) = \sum_{k=1}^{n-1} T(k) + 1$

b) $T(n) = \sum_{k=1}^{n-1} T(k) + 7$

c) $T(n) = \sum_{k=1}^{n-1} T(k) + n^2$

d) $T(n) = 2\sum_{k=1}^{n-1} T(k) + 1$

Určete řád růstu funkce $T(n)$.