1. príklad

max:=1

je max < 8 ? ANO

max:=8

je max <12 ? ANO

max:=12

je max <9 ? NIE

je max <11 ? NIE

je max <2 ? NIE

je max <14 ? ANO

max:=14

je max < 5 ? NIE

MAXIMUM je teda 14

2. príklad

a)procedure zdvojnásob (n : kladné celé číslo)

while n >= 0

n := 2n

VLASTNOSTI:

Input a output MÁ

Jednoznačnosť MÁ

Konečnosť NEMÁ (podmienka vo while bude vždy splnená)

Správnosť NEMÁ (podľa názvu procedúry má n prenásobiť dvomi, čo sa

ale stane nespočetne veľa krát)

Všeobecnosť MÁ

b)procedure vyber (a , b : celé čísla)

x := a alebo b

Input a output MÁ

Jednoznačnosť NEMÁ (zo pseudokódu nie je jasné podľa akého kritéria má algoritmus

vybrať jedno z dvoch vstupných parametrov)

Konečnosť MÁ

Správnosť MÁ (algoritmus priradí x jednu z dvoch hodnôt aj keď nie je jasné ako)

Všeobecnosť MÁ

c) procedure vydeľ(n: kladné celé číslo)

while n>=0

begin

m:=1/n

n=n-1

end

Input a output MÁ

Jednoznačnosť MÁ

Konečnosť NEMÁ (n bude vždy >=0)

Správnosť NEMÁ (v predposlednom riadku porovnávať n a n-1 nemá zmysel)

Všeobecnosť MÁ

d)procedure súčet (n : kladné celé číslo)

súčet := 0

while i < 10

súčet :=súčet + i

Input a output MÁ

Jednoznačnosť MÁ

Konečnosť MÁ (len ak sa hodnota premennej i mení mimo tohto cyklu pomocou iného procesu alebo cyklu)

Správnosť NEMÁ (vzhľadom k tomu že nie je použitý vstupný parameter)

Všeobecnosť NEMÁ (nepoužíva vstupný parameter nedá sa použiť s rôznymi premennými)

3. príklad

1.do premennej sucet priradím prvý prvok postupnosti

2.vo for cykle pripočítavam nasledujúci prvok postupnosti k premennej sucet až kým

nepripočítam posledný n-tý prvok

procedure spočítajPostupnosť(a1,a2...an:postupnosť čísel)

sucet:=a1

for i:=2 to n

sucet:=sucet+ai

{výsledok je v premennej sucet}

Časová náročnosť:

sucet:=a1

for i:=2 to n -- (n-2)x zvýšení hodnoty premennej i (teda i:=i+1)

sucet:=sucet+ai -- (n-1)x bitových operácií sčítania

--celková časová náročnosť je (n-2)+(n-1)

4. príklad

procedure najväčšíRozdiel(a1,a2...an: postupnosť čísel)

rozdiel:=|a1-a2|

for i:=3 to n

rozdielPom:=|ai-a(i-1)|

if rozdielPom>rozdiel then rozdiel:=rozdielPom

{výsledok je v premennej rozdielPom}

Časová náročnosť:

rozdiel:=|a1-a2| -- 1x bitová operácia odčítania

for i:=3 to n -- (n-3)x operácia zvýšenia hodnoty premennej i (teda i:=i+1)

rozdielPom:=|ai-a(i-1)| -- 2\*(n-3)x operácia odčítania(premenné a index)

if rozdielPom>rozdiel then rozdiel:=rozdielPom

--celková časová náročnosť je 1+(n-3)+2\*(n-3)

--|| je absolútna hodnota medzi bitové operácie pri výpočte časovej zložitosti nebola započítaná

5. príklad

Slovne:

Prejdem všetky prvky neklesajúcej postupnosti čísel od druhého prvku až po n-tý

prvok, pričom vždy zisťujem rozdiel i-tého a (i-1)-vého prvku. Ak je rozdiel 0 sú to rovnaké prvky

pridám do poľa c tento prvok až po zistení či (i-1) a (i-2) prvok postupnosti sú nerovnaké.

Tým zabezpečím aby sa prvok pridal do postupnosti len vtedy ak je pred ním iné číslo ako on samotný.

Pri prvých dvoch prvkoch(i=2) to nie je potrebné zisťovať a prvok sa hneď uloží do poľa.

procedure opakovanieHodnôt(a1,a2...an:neklesajúca postupnosť čísel)

for i:=2 to n

if (ai-a(i-1))=0 then

if (i!=2)then

if(a(i-1)-a(i-2))!=0 then

pridaj ai do pola c

else

pridaj ai do pola c

{postupnosť c obsahuje čísla, ktoré sa opakujú v postupnosti a1 (každé opakujúce číslo jeden krát) }

Časová náročnosť:

for i:=2 to n --(n-2)x zvyšovanie hodnoty premennej i

if (ai-a(i-1))=0 then --2\*(n-2) bitových operácií odčítania (i-1 a následne prvky medzi sebou)

if (i!=2)then

if(a(i-1)-a(i-2))!=0 then --3\*(n-2) bitových operácií odčítania

pridaj ai do pola c

else

pridaj ai do pola c

--celková časová náročnosť 6\*(n-2)

6. príklad

postupnosť 1, 1, 1, 1, 2, 3, 3, 4, 5

je (1-1) rovné 0? ÁNO

je i=2? ÁNO

pridaj 1 do pola c

je (1-1) rovné 0? ÁNO

je i=2? NIE

je (1-1)!=0? NIE

je (1-1) rovné 0? ÁNO

je i=2? NIE

je (1-1)!=0? NIE

je (1-2) rovné 0? NIE

je (3-2) rovné 0? NIE

je (3-3) rovné 0? ÁNO

je i=2? NIE

je (3-2)!=0? ÁNO

pridaj 3 do pola c

je (4-3) rovné 0? NIE

je (5-4) rovné 0? NIE

7. príklad

Vstupom do algoritmu je slovo dĺžky n(párne alebo nepárne kladné celé číslo).

Ak je n párne číslo, slovo predelím na dve časti, jednú časť otočím a porovnám či sa

tieto časti zhodujú.

Ak n je nepárne číslo vypočítam k:=└n/2┘ následne zo vstupného slova odrežem prvých a posledných k prvkov jednu časť otočím a porovnám ich či sa rovnajú.

procedure zistiPalindrom(slovo: String retazec)

je:=0 {0 nie je palindrom, 1 je palindrom}

n:= dĺžka reťazca

if (n mod 2) =0 then

cast1:=(písmena v slove na indexoch 1 až n+1)

cast2:=(písmena v slove na indexoch n+2 až n)

cast2:= otoč cast2

if cast1=cast2 then je :=1

else

k:=└n/2┘

cast1:=(písmena v slove na indexoch 1 až k+1)

cast2:=(písmena v slove na indexoch k+2 až n)

cast2:= otoč cast2

if cast1=cast2 then je:=1

{je=1-je palindrom , je=0 nie je palindrom}

9. príklad

procedure nasobenieMatic(A : matica veľkosti ij, B : matica veľkosti ij)

for riadky := 1 to i

for stlpce:=1 to j

for r := 1 to i

sum := sum + A[riadky][r]\*B[r][stlpce];

multiply[riadky][stlpce] = sum;

sum = 0;

{matica multiply obsahuje výsledok násobenia matíc A a B}

Časová náročnosť algoritmu:

for riadky := 1 to i --(i-1)x zvýšenie hodnoty premennej riadky

for stlpce:=1 to j --((i-1)\*(j-1))x zvýšenie hodnoty premennej stlpce

for r := 1 to i --((i-1)\*(j-1)\*(i-1))x zvýšenie hodnoty r

sum := sum + A[riadky][r]\*B[r][stlpce]; --2\*(i-1)\*(j-1)\*(i-1)sú dve operácie

multiply[riadky][stlpce] = sum;

sum = 0;

--celková časová náročnosť algoritmu (i-1)+((i-1)\*(j-1))+3\*((i-1)\*(j-1)\*(i-1))

9. príklad

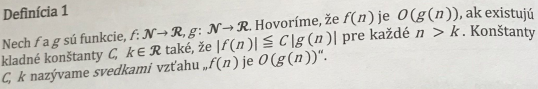
*f* (*n* ) je *O* (*g*(*n* )) má vlastnosti:

tranzitivnosť

reflexívnosť?

f(n) je

Symetrickosť je jasne vysvetlená v samotnej definícii :



Tranzitívnosť:

Nech f(n) je a g(n) je tak existujú konštanty C1,k1 ;C2,k2 ;C3,k3 ;C4,k4 také že:

|f(n)| <=C1\*|g(n)| pre každé n>k1

|g(n)| <=C2\*|f(n)| pre každé n>k2

|g(n)| <=C3\*|h(n)| pre každé n>k3

|h(n)| <=C4\*|g(n)| pre každé n>k4

Z toho vyplýva že:

|f(n)| <=C1\*|g(n)| <=C1\*C3\*|h(n)|

|f(n)| <= C1\*C3\*|h(n)|

A tiež vyplýva:

|h(n)| <=C4\*|g(n)| <=C4\*C2\*|f(n)|

|h(n)|<=C4\*C2\*|f(n)|

Dôkaz reflexívnosti:

10. príklad

a) f(x)=17x+11

1<x< pre každé x>1

Výsledok: C=28 a K=1

b)f(x)=

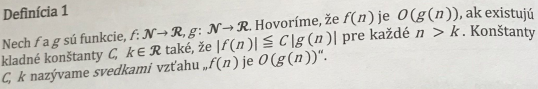
c)f(x)=

Výsledok:

Vypíšme si niekoľko členov funkcie

(limita pre x idúce do nekonečna by bola nekonečno)

Z týchto členov vidíme, že postupnosť po prvých pár členoch začne rásť.

Nie je teda možné nájsť také C pre ktoré by vzťah 

Platil pre každé n>k.

d)

nech t= (celá časť čísla) a d je desatinná časť čísla

Výsledok:

v závislosti od parametru d môže byť malé aj veľké číslo. Zanedbajme túto časť

Stačí C zvoliť =2

Vidíme že tento vzťah platí vždy. Svedkami vzťahu sú teda C=2 a k=0

11. príklad

1. *n*

Musí platiť |e(n)|<=C\*|f(n)|

Zvoľme C=1 a n=1 000 000 (log počítam so základom e nie 2)

1. 13815524373510,558
2. 1327197370780000

Výsledok je teda a),c),b) kde svedkami vzťahu sú napríklad C=1 a k =1000000

12. príklad

