Aplikovaná informatika

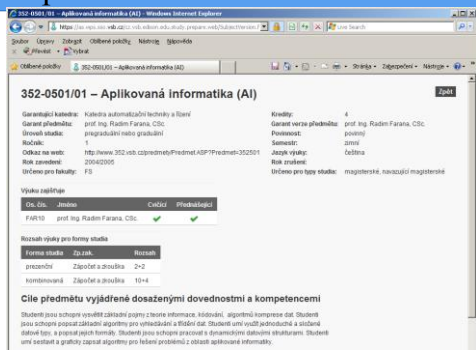
Podklady předmětu
Aplikovaná informatika
pro akademický rok 2013/2014
Radim Farana

1

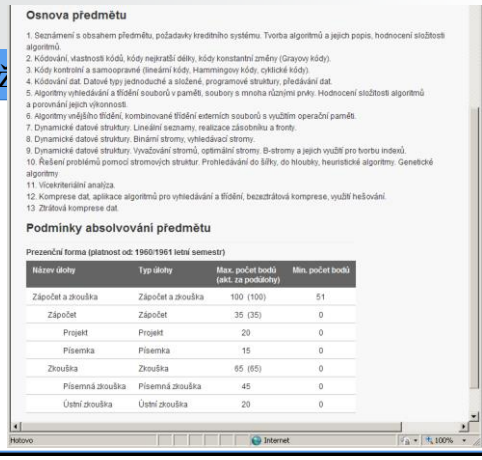
Obsah

- Obsah předmětu,
- Požadavky kreditového systému,

Obsah předmětu



Pož



Osnova předmětu

1. Seznámení s obsahem předmětu, požadavky kreditního systému. Tvorba algoritmů a jejich popis, hodnocení složitosti algoritmů.
2. Kódování, vlastnosti kódů, kódy nejkratší délky, kódy konstantní změny (Grayovy kódy).
3. Kódy kontrolní a samoopravné (lineární kódy, Hammingovy kódy, cyklické kódy).
4. Kódování dat. Datové typy jednoduché a složené, programové struktury, předávání dat.
5. Algoritmy vyhledávání a třídění souborů v paměti, soubory s mnoha různými prvky. Hodnocení složitosti algoritmů a porovnání jejich výkonnosti.
6. Algoritmy vnějšího třídění, kombinované třídění externích souborů s využitím operační paměti.
7. Dynamické datové struktury. Lineární seznamy, realizace zásobníku a fronty.
8. Dynamické datové struktury. Binární stromy, vyhledávací stromy.
9. Dynamické datové struktury. Vyvažování stromů, optimální stromy. B-stromy a jejich využití pro tvorbu indexů.
10. Řešení problémů pomocí stromových struktur. Prohledávání do šířky, do hloubky, heuristické algoritmy. Genetické algoritmy.
11. Vícekriteriální analýza.
12. Komprese dat, aplikace algoritmů pro vyhledávání a třídění, bezeztrátová komprese, využití hešování.
13. Ztrátová komprese dat.

Podmínky absolvování předmětu

Prezenční forma (platnost od: 1960/1961 letní semestr)

Název úlohy	Typ úlohy	Max. počet bodů (akt. za podúlohy)	Min. počet bodů
Zápočet a zkouška	Zápočet a zkouška	100 (100)	51
Zápočet	Zápočet	35 (35)	0
Projekt	Projekt	20	0
Písemka	Písemka	15	0
Zkouška	Zkouška	65 (65)	0
Písemná zkouška	Písemná zkouška	45	0
Ústní zkouška	Ústní zkouška	20	0

Obsah

- Tvorba algoritmů,
  - vlastnosti algoritmu.
- Popis algoritmů,
  - vývojové diagramy,
  - diagramy aktivit,
  - strukturogramy.
- Hodnocení složitosti algoritmů,
  - vypočitatelnost,
  - časová složitost,
  - NP a NP-úplné problémy.

Algoritmus

- Algoritmus je přesný předpis definující výpočtový proces vedoucí od měnitelných výchozích údajů až k žádaným (vždy správným) výsledkům. Tento předpis se skládá z jednotlivých výpočtových kroků, které jsou zapsány v určitém pořadí. Počet výpočtových kroků musí být konečný.

Vlastnosti algoritmu

- **determinovanost** - shrnuje přesnost, srozumitelnost a jednoznačnost. V každém okamžiku řešení musí být jasné, jakou operaci má algoritmus provádět.
- **hromadnost** (masovost) - algoritmus musí popisovat zpracování celé skupiny příbuzných hodnot.
- **rezultativnost** - algoritmus musí vždy dospět ke správnému výsledku, a to pomocí konečného počtu kroků.
- **opakovatelnost** - při stejných hodnotách vstupních dat musí algoritmus vždy dospět ke stejnému výsledku.

Algoritmus versus program

- program = posloupnost příkazů,
  - dokumentuje se výpisem programu,
  - je chráněn autorským zákonem.
- algoritmus = postup práce,
  - dokumentuje se zápisem algoritmu,
  - je možné ho patentovat.
- Program realizuje algoritmus (algoritmy), algoritmus je jeho nutnou součástí.

Popis algoritmu

- Slovní popis
  - pracovní postup,
  - strukturovaný text, zápis pomocí grafu,
  - pseudokód (programovací).
- Grafický zápis
  - vývojový diagram,
  - Kopenogram, NS-diagram,
  - strukturogram.

Vývojový diagram

- Popis algoritmů pro FORTRAN (FORmula TRANslator)
- IBM v r. 1954
- Formalizován různými normativy (ČSN 36 9030)

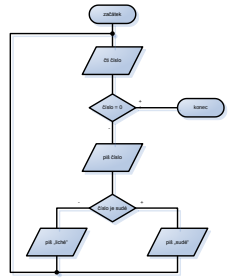
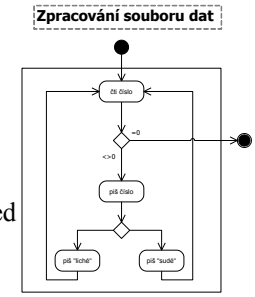


Diagram aktivit UML

- Diagramy aktivit se podobají vývojovým diagramům, ale jsou nástrojem komplexního CASE (Computer-Aided Software Engineering) systému s názvem Unified Modeling Language, ve zkratce UML.

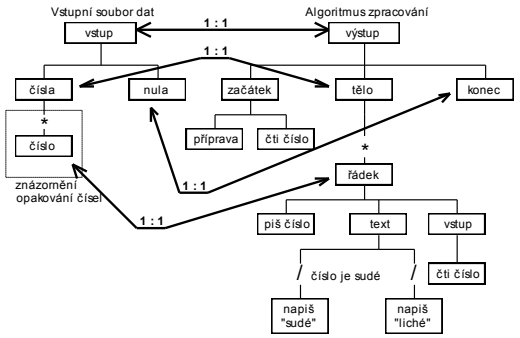


Strukturogram

Michael Anthony Jackson
* 1936
http://mcs.open.ac.uk/mnj665/index.html

- Michael A. Jackson, 1975
- Základní struktury:
  - sekvence (posloupnost operací),
  - selekce (větvení).
  - opakování – zvláštní případ sekvence.
- Snadné postupné upřesňování algoritmu
- Jednoznačný vztah mezi daty a algoritmem

Strukturogram



Základní struktury

Pseudojazyk	Vývojový diagram	Diagram aktivit	Strukturogram
Programový celek (rutina, podrogram, procedura a pod.) - definice			

`název (parametry)`
`.`
`.`
`.`
`end název`
	začátek
	⋮
	konec
			jeden celý strukturogram

programový celek - použití (volání)			
`název (parametry)`	název(par.)	název (par.)	název(par.)

Podmíněná činnost, rozhodování

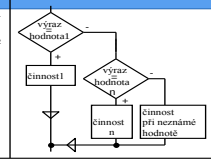
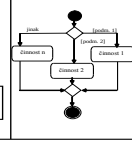
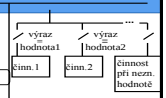
Pseudojazyk	Vývojový diagram	Diagram aktivit	Strukturogram
Provedení konkrétní činnosti			
`popis činosti`	popis činnosti	popis činnosti	popis činnosti
Podmíněná činnost (provádí se pouze pokud je splněna určitá podmínka)			

`if podmínka`
`  podmíněná činnost`
`end if`
	podmínka
	-
	+
	činnost
		smysl má jen rozhodování	podmínka
			činnost

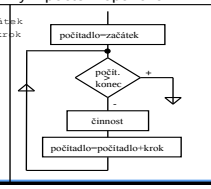
Rozhodování (pokud platí určená podmínka, provede se činnost 1, jinak činnost 2)			

`if podmínka`
`  činnost 1`
`else`
`  činnost 2`
`end if`
	podmínka
	+
	-
	činnost 1
	činnost 2
			podmínka
			činnost 1
			činnost 2

Pseudojazyk	Vývojový diagram	Diagram aktivit	Strukturogram
Větvení (podle hodnoty výrazu se provádí určená činnost)			

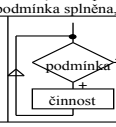
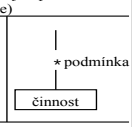
`case výraz=hodnota 1`
`  činnost 1`
`case výraz=hodnota 2`
`  činnost 2`
`.....`
`case else`
`  činnost při neznámé hodnotě`
`end case`
			

Opakování s pevným počtem opakování			

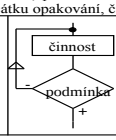
`for počítadlo=začátek`
`  to konec step krok`
`  činnost`
`end for`
		je nutno sestavit z ostatních značek podobně jako u vývojového diagramu	

Opakování

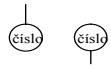
Pseudojazyk	Vývojový diagram	Diagram aktivit	Strukturogram
Opakování s testem na začátku (dokud platí podmínka, činnost se opakuje - pokud na začátku opakování není podmínka splněna, činnost se vůbec neprovede)			

`while podmínka`
`  činnost`
`end while`
		je nutno sestavit z ostatních značek podobně jako u vývojového diagramu	

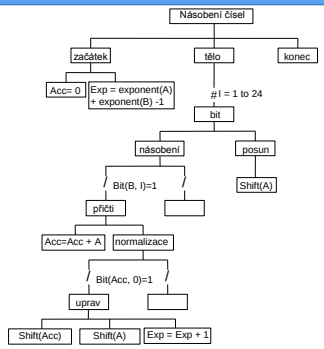
Opakování s testem na konci (opakování končí, pokud je splněna podmínka - i když podmínka platí již na začátku opakování, činnost se jednou provede)			

`repeat`
`  činnost`
`until podmínka`
 	
(v některých jazycích
`while`
`  činnost`
`end while podm.)`
		je nutno sestavit z ostatních značek podobně jako u vývojového diagramu	neexistuje

Zvláštní činnosti

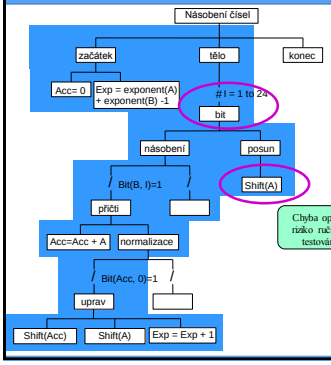
Pseudojazyk	Vývojový diagram	Diagram aktivit	Strukturogram
Vstupní nebo výstupní operace			
`Jako každá` `jiná činnost`		jako každá jiná aktivita	Jako každá jiná činnost
Přípravná činnost			
`Jako každá` `jiná činnost`		jako každá jiná aktivita	Jako každá jiná činnost
Spojka (činnost končí v jedné části algoritmu a pokračuje v jiné části)			
`Neexistuje`		Neexistuje	Neexistuje

Algoritmus násobení dvou čísel Single



Proměnné:
A, B – činitelé
(normalizované mantisy)
Acc – akumulátor
Exp – exponent
I – celočíselné počítadlo
Funkce:
Shift(x) – bitová rotace vpravo
exponent(x) – exponent čísla

Testování algoritmu



Proměnné:
A : 0,11001100110011001100110011100 . 2^{-3}
0,01100110011001100110011001110
0,00110011001100110011001110011
0,00011001100110011001100111001

B : 0,10001100110011001100110011100 . 2^{-3}

Acc : ~~0~~
1,00110011001100110011001110010
1,00110011001100110011001110010

Chyba opisu, rizika načítání testování

Exp: ~~7~~ -6

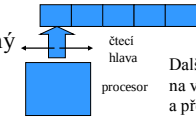
I: ~~1~~ ~~2~~ 3

atd.

Hodnocení složitosti algoritmů

- Vypočitatelnost algoritmu
  - Turingův stroj (Alan M. Turing, 1936) – abstraktní model počítače,
  - s Alonzem Churchem vyslovili domněnku, že je ekvivalentní s počítačem (s nekonečnou pamětí),
  - Dokázal, že nelze sestavit stroj, který určí, zda se libovolný stroj zastaví.

Deterministický Turingův stroj



Další krok závisí na vnitřním stavu a přečtené hodnotě

Časová složitost algoritmu

- Čas na vykonání algoritmu vyjádřený v počtu elementárních operací.
- Všechny operace trvají stejně dlouho.
- Složitost algoritmu je funkcí vstupu (n).
- Konstantní rozdíly se ignorují.
- Nechť $f(n)$, $g(n)$ jsou funkce z množiny přirozených čísel do množiny reálných čísel, pak $f(n) = O(g(n))$, pokud existuje konstanta c, že pro velké n platí: $f(n) \leq cg(n)$.

Výpočet složitosti algoritmu

Násobení dvou čtvercových matic rozměru (n n)

```
Sub NasobeniCtvercovychMatic(a(n n),b(n n),c(n n))
    For I = 1 to n                          → n *(2+
        For J = 1 to n                      → n *(2+
            c(I, J) = 0                     → 1+
            For K = 1 to n                  → n *(2+
                c(I, J) = c(I, J) +         → 2
                          a(I, K)*b(K, J)
            Next K                          → )
        Next J                              → )
    Next I                                  → )
End Sub
```

$f(n) = n(2+n(2+1+n(2+2)))=4n^3+3n^2+2n$

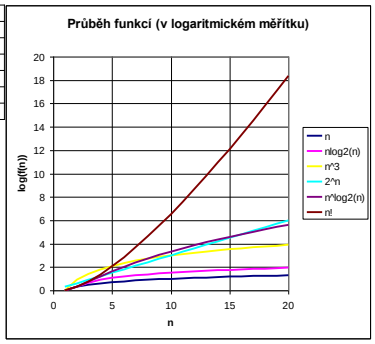
zjednodušení $f(n) = n(\ n(\ \ n(\ 1 \ )))= n^3$

Složitost algoritmů

- Polynomiální složitost: výpočet vyžaduje $O(k^d)$ bitových operací
  - sčítání $d = 1$,
  - násobení $d = 2$.
- Nepolynomiální složitost, složitost výpočtu s rostoucím n roste rychleji.
  - $n!$ - $O(n\log_2^2(n)) = O(2^k k^2)$.
  - a^n

Příklady funkcí

Funkce	Přibližné hodnoty		
n	10	100	1000
$n\log_2(n)$	33	664	9966
n^3	1000	1000000	1000000000
2^n	1024	1,27E+30	1,07E+301
$n^{\log_2(n)}$	2099	1,94E+13	7,90E+29
$n!$	3628800	9,33E+157	4E+2567



Třídy NP a NP-úplných problémů

- P – schůdné algoritmy běžící nejhůře v polynomiálním čase (PT) – násobení,
- NP problém (nondeterministic polynomial) – neschůdné algoritmy, v PT lze pouze ověřit správnost řešení – faktorizace ,
- NP-úplný problém – NP problémy vzájemně mapovatelné v PT – obchodní cestující, (existuje P transformace jednoho na druhý).