

SEMESTRÁLKY

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)



30.1.2015

I. Opravný termín

AKA Honzík's killing spree

LEGENDA:

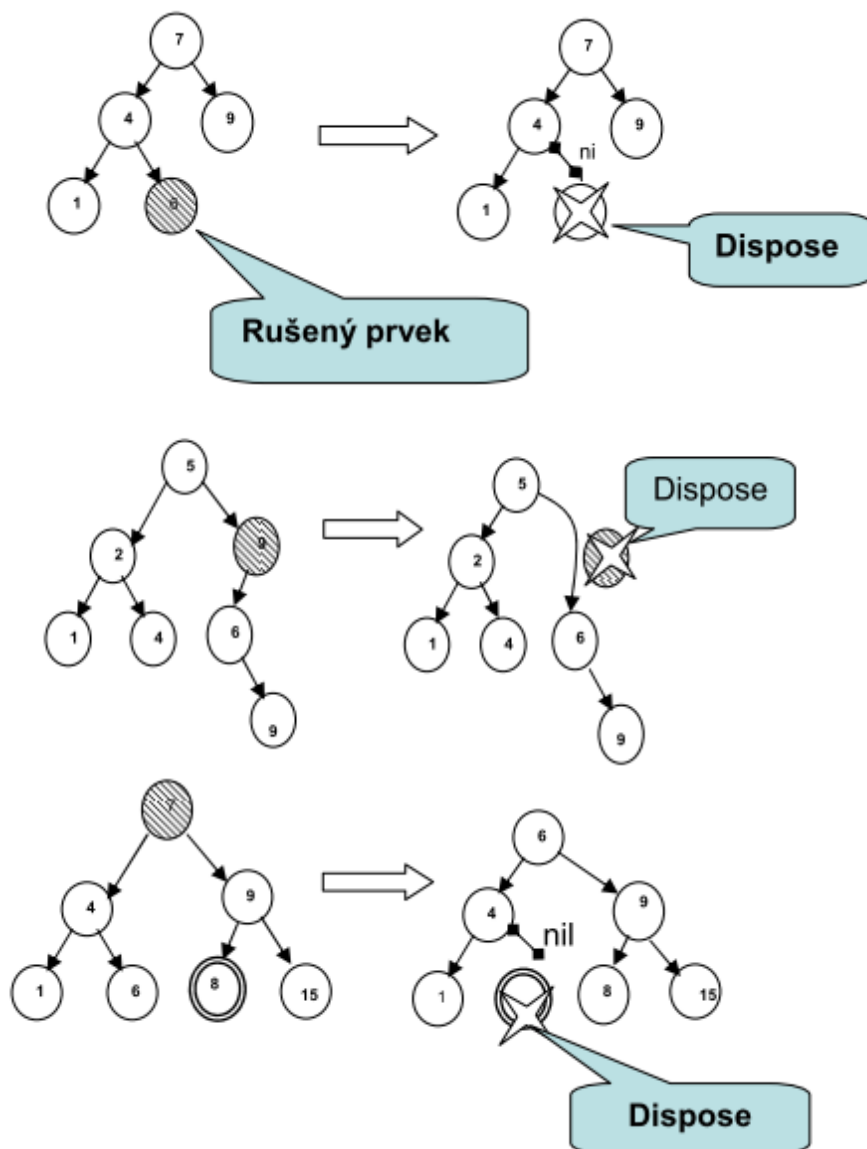
ODPOVED: nevyplnené, treba doplniť

ODPOVED: nejasnosti, treba overiť

ODPOVED: vypracované, ale môžete skontrolovať :)

Zadaný BVS a nakreslit ho po odstranění 3 prvků.

odpoved:



(pozn. Jedná se o návod. Nikoliv o příklad)

LEGENDA:


ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Listmerge sort

odpoved:



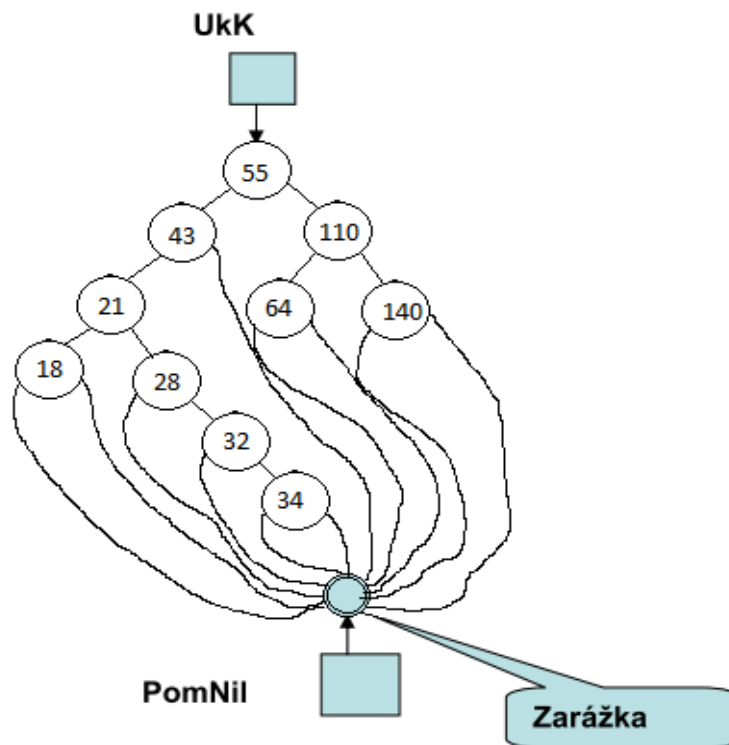
Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	2	5	9	3	7	10	16	5	8	6	3	11	18	20
Pom	2	3	0	5	6	7	0	9	0	0	12	13	14	0

(pozn. Jedná převzato z opory. Příklad měl stejný princip, jiná čísla)

BVS se zarážkou

naplnit hodnotami (55, 110, 64, 140, 43, 21, 28, 32, 34, 18)

odpoved:



LEGENDA:

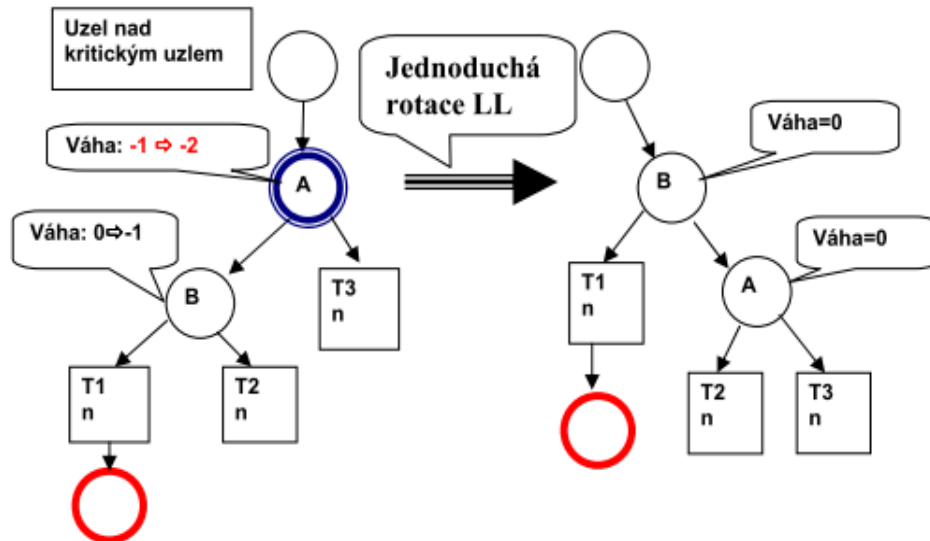
ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale možte skontrolovat' :)

LL rotace AVL

odpoved:



(pozn. Jedná se obrázek ze skript. Skutečný příklad byl velice podobný, tuším zrcadlový)

Definice ekvivalence seznamu

odpoved:

Rekurzivní definice: Dva seznamy jsou ekvivalentní, když jsou oba prázdné a nebo když se rovnají jejich první prvky a současně jejich zbývající části seznamu.

Popsat jakým způsobem smažeme prvek za aktivním v jednosměrném seznamu bez použití cyklu

odpoved:

Postdelete.

(pozn. Myslím, že zadání bylo trochu jiné. Toto moc nedává smysl)

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Doplnění kódu nerekurzivního pre/in/post order

odpoved:

Procedure NejlevPre/NejlevIn(Uk:TUk; var L:TList);

```
begin
    while Uk <> nil do
        begin
            Push(S, Uk);
            InsertLast(L, Uk^.Data); // PREORDER
            Uk := Uk^.LUk
        end
    end;
```

Procedure PreOrder (Uk:TUk; var L:TList);

```
begin
    ListInit(L);
    SInit(S);
    NejlevPre(Uk, L);
    while not SEmpty(S) do
        begin
            Top(S, Uk);
            Pop(S);
            NelvejPre(Uk^.PUk, L)
            InsertLast(L, Uk^.Data); // INORDER
        end
    end;
```

Procedure NejlevPost (Uk:TUk; var L:TList);

```
begin
    while Uk <> nil do
        begin
            Push(S, Uk);
            Push(SB, true);
            Uk:= Uk^.LUk
        end
    end;
```

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Procedure PostOrder (Uk:TUk; var L:TList);

```
var
    Zleva:Boolean;
begin
    ListInit(L);
    SInit(S);
    SInitB(SB);
    NejlevPost(Uk);
    while not SEmpty(S) do
        begin
            Top(S, Uk);
            TopB(SB, Zleva);
            PopB(SB);
            if Zleva then
                begin
                    PushB(SB, false);
                    NejlevPost(Uk^.PUK)
                end
            else
                begin
                    Pop(S);
                    InsertLast(L, Uk^.Data)
                end
            end
        end
    end;
end;
```

(pozn. Jednalo se o jejich spojení v jednom programu, ale dávalo to smysl v případě znalosti výše zmíněných)

Dopsat kód fronty (REMOVE)

Procedure Remove(var Q:TQueue);

```
begin
    if Q.QZac <> Q.QKon then
        begin
            Q.QZac := Q.QZac + 1;
            if Q.QZac > Q.QMax then
                Q.QZac := 1
            end
        end
    end;
end;
```

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

20.1.2015
Řádný termín
AKA Surprise butt sex

Doplnit kód pro rekurzivní ekvivalenci struktur dvou binárních stromů. (8 b)

odpoved:

```
function EQTS(Kor1, Kor2:TUk): Boolean;
begin
    if (Kor1=nil) or (Kor2=nil) then
        EQTS := (Kor1 <> nil) and (Kor2 <> nil)
    else
        EQTS := EQTS(Kor1^.Luk, Kor2^.Luk) and EQTS (Kor1^.Puk, Kor2^.Puk)
        (*and (Kor1^.Data = Kor2^.Data) *)
end;
```

**v případě ekvivalence dvou stromů,
což není to samé jako ekvivalence dvou struktur.**

Partition řádek Quicksort + index i, j (5+3 b)

odpoved:

<i>i:</i>	1	2	3	4	5	6	7	8	9	10
pred:	6	3	7	9	4	0	1	5	8	2
po:	2	3	1	0	4	9	7	5	8	6

i:	6
j:	4

Rekurzivní definice délky lineárního seznamu (3 b)

odpoved:

Seznam má délku nula pokud je prázdný, jinak má délku jedna plus délka zbytku
seznamu

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Shell Sort , přepsat aby se nepřesouvaly hodnoty, ale indexy v poli

Por[] (8 b)

odpoved:

$A[j] > A[j+step] \rightarrow A[Por[j]] > A[Por[j+step]]$

$A[j] := A[j+step] \rightarrow Por[j] := Por[j+step]$

Popsat vkládání prvku před prvek, jednosměrný seznam (3 b)

odpoved:

Vložíme nový prvek metodou PostInsert za aktivní prvek. Vezmeme hodnotu aktivního prvku a zapíšeme ji do nového prvku. Hodnotu aktivního prvku přepíšeme novými daty a posuneme aktivitu (Succ)

Čím se liší BS se zpětnými ukazateli od normálního BS a implementace jakých operací se liší od implementace operací v normálním BS (4 b)

odpoved:

BS se zpětnými ukazateli používáme když se chceme při průchodu INORDER vyhnout rekurzi nebo použití zásobníku.

Zpětný uzel kořene ukazuje na NIL (všechny uzly vedlejší diagonály takéž)

Zpětný ukazatel levého syna ukazuje na svého otce.

Zpětný ukazatel pravého syna ukazuje tam kam otec.

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

TRP 2HASH + Brentova varianta (indexovane od 0) (6+6 b)

odpoved:

Bylo třeba doplnit číslo 59

i:	0	1	2	3	4	5	6	7	8	9	10
dat:	xx	xx	xx	xx	15			xx	19		
2HASH	xx	xx	xx	xx	15			xx	19		59
BRENT	xx	xx	xx	xx	59			xx	19		15

- Přečísľuju si (stačí si mi pamatovat dva vzorce) takže indexy jsou teď od 1..11
- Pro vložení X použiju vzorec: $(X \text{ Mod } \text{MAX}) + 1 = \text{indexPozice}$
- Pro posun X použiju vzorec: $(X \text{ Mod } (\text{MAX} - 1)) + 1 = \text{velikostKroku}$

2HASH

59 Mod 11 + 1 = 5 // na mém (přečísľovaném) indexu 5 už leží číslo 15, takže si vypočítám krok.

$(11 - 1) + 1 = 159 \text{ Mod } 0$ // Počítám 10 kroků od indexu 4, skončím na 3 kde je číslo, tak zkusím od 3 dalších 10 kroků atp... skončím po 5 na indexu 10 + 1 krok (vložení)

BRENT

Počítám obdobně, jen když zjistím, že 59 nejde vložit zkusím krok a ten vede za hranici (větší než 10 index + viděli jsme kolik je třeba kroků na uložení 59 za pomocí 2HASH metody) takže 59 uložím na místo 15 a pokusím se s ní udělat krok

$15 \text{ Mod } (11 - 1) + 1 = 6$ // Padne na 10 index a mám hotovo

KROK 2hash	6
------------	---

Mnohacestné seřídování - 1000 souborů, 8 pásek, napsat vzorec a zaokrouhlit výsledek (4 b)

odpoved:

$$2K = 8$$

$$K = 4$$

$$N = 1000 \quad \log_k N \Rightarrow \log_4 1000 \Rightarrow 4^x = 1000$$

vzorec:

$$x = 4.9 = 5 \text{ (zaokrouhleně)}$$

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Opravný termín 2014

Doplnění kódu PostDelete u jednosměrně vázaného seznamu.

(8 bodů)

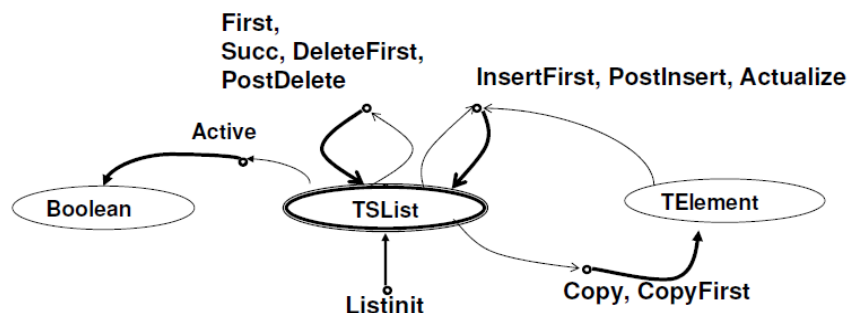
odpoved:

```
Procedure PostDelete(var L:TList);
var
    PomUk:Tuk;
begin
    if (L.Act<>nil) then
        begin
            if L.Act^.UkNasl <> nil then
                begin
                    PomUk:=L.Act^.UkNasl;
                    L.Act^.UkNasl:=PomUk^.UkNasl;
                    Dispose(PomUk)
                end
            end
        end
end;
```

Diagram signatury dvousměrně vázaného seznamu. (4 body)

odpoved:

Diagram signatury ADT TList



Symetricky doplňkové operace pro dvousměrný seznam:
Last, Pred, DeleteFirst, PostDelete, InsertLast, PreInsert, CopyLast

12.9.2014

3. přednáška

14

Operace co chybí v seznamu pro dvousměrný seznam: DeleteLast, PreDelete

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

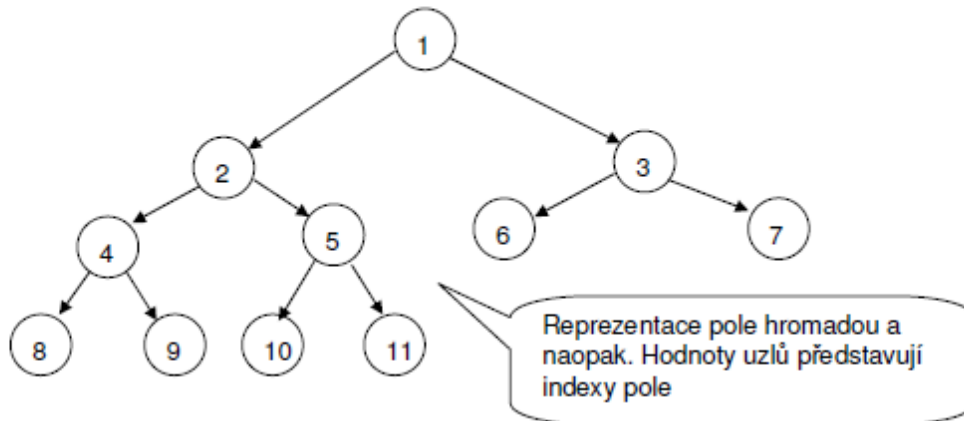
ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale možte skontrolovat' :)

X Prvkové pole (hromada) v pořadí kde prvky se shodují s indexy
tvorba stromu který hromada představuje
průchody PreOrder, InOrder a PostOrder

odpoved:

1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	----	----



14 E

PreOrder: 1,2,4,8,9,5,10,11,3,6,7
PostOrder: 8,9,4,10,11,5,2,6,7,3,1
InOrder: 8,4,9,2,10,5,11,1,6,3,7
InvPreOrder: 1,3,7,6,2,5,11,10,4,9,8
InvInOrder: 7,3,6,1,11,5,10,2,9,4,8
InvPostOrder: 7,6,3,11,10,5,9,8,4,2,1

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale možte skontrolovat' :)

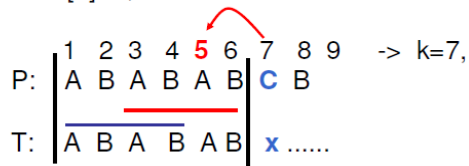
KMP – Vytvořit vektor Fail řetězce ABCABCAD (4 body)

Konstrukce automatu KMP:

Automat je reprezentován vzorkem P a vektorem FAIL, který má prvky typu integer, reprezentující cílový index zpětné šipky.

Příklad tvorby vektoru pro P='ABABABCB'

FAIL[1]=0;



Předpokládejme, že $x \neq C$; pak další možné místo, na kterém může vzorek v textu začínat je třetí pozice. Protože došlo k nesouhlasu na 7 pozici a protože platí:

$(P_1 \dots P_4) = (P_3 \dots P_6)$ může nové porovnání začít na indexu 5 a tedy $FAIL[7]=5$

Platí tedy: FAIL: 0 1 1 2 3 4 5 1

$FAIL[k] = \max r \{ (r < k) \text{ and } (P_1 \dots P_{r-1}) = (P_{k-r+1} \dots P_{k-1}) \}$

12.9.2014

21

odpověď:

0 1 1 2 3 4 5 1

ABCABCA|D \rightarrow 0 1 1 1 2 3 4 5 - Fail[8] = 5

ABCABC|AD \rightarrow 0 1 1 1 2 3 4 5 - Fail[7] = 4

Radix sort, čím se liší první průchod od ostatních (3 body)

odpověď:

V prvom cykle prechádza pole podľa indexu, v ostatných podľa ukazatelu

Infix to Postfix/Prefix nějakého výrazu. (6 bodů)

Infix \rightarrow PostFix:

odpověď:

$(a+b) \cdot (c-d) / (e+f) \cdot (g-h) \Rightarrow ab+cd-*ef+/gh-*$

Infix \rightarrow PreFix:

odpověď:

$(a+b) \cdot (c-d) / (e+f) \cdot (g-h) \Rightarrow */*+ab-cd+ef-gh$

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Doplnit kód rekurzívneho zápisu CopyTree. (10 bodů)

odpoved:

```
procedure CopyTree(KorOrig:TUk; var KorCopy:TUk);
begin
    if KorOrig <> nil then
    begin
        new(KorCopy);
        KorCopy^.Data := KorOrig^.Data;
        CopyTree(KorOrig^LUk, KorCopy^.LUk);
        CopyTree(KorOrig^PUk, KorCopy^.PUk);
    end
    else
        KorCopy:=nil
end;
```

Doplnit kód funkcí u fronty QFull a QRemove (10 bodů)

odpoved:

```
function QFull(Q: TQueue): Boolean;
begin
    QFull:= (Q.Zac=1) and (Q.Kon=QMax) or
    ((Q.Zac - 1) = Q.Kon)
end;
```

```
procedure QRemove (var Q:TQueue);
begin
    if Q.QZac<>Q.QKon then
    begin
        Q.QZac:=Q.Qzac + 1;
        if Q.QZac > Q.QMax then
            Q.QZac:=1;
        end
    end
end;
```

LEGENDA:

ODPOVED: nevyplnené, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

Řádný termín 2014

skupina A

Partition řádek u QuickSortu a napsat indexy i, j kde skončily (6+4b)

odpoved:

pseudo median vypocitame tak ze spocitame Index zaciatku a Index konca na vysledok pouzijeme operaci DIV 2 a vyde nam Index ktoreho hodna je nas hladany PSEUDOMEDIAN
cize: $(1+10)\text{div}2 = 5$ Index 5 zodpoveda hodnote 4.

PM=pseudomedian

postup:

1. zaciname s $i=1$ a $j=n$ (n je v nasom pripade $10 = \text{max index}$)
2. v i hladame take ktore je vacsie ako PM ak take najdeme tak ho budeme menit
3. v j hladame take ktore je mensie ako PM ak take najdeme tak ho budeme menit
4. ak v bodoch 2 a 3 take cisla nenajdeme tak i inkrementujeme a j dekrementujeme o 1
5. ak take najdeme tak ich zamenime (vid 1-2 riadok DATA pre cisla 6 a 1) **po zamene nesmieme zabudnut inkrementovat i a dekrementovat j .**
6. postup 2 az 5 opakujem dokym $i > j$

Pseudomedián=4

Index	1	2	3	4	5	6	7	8	9	10
Data	6	2	7	9	4	10	3	5	1	8
	1	2	7	9	4	10	3	5	6	8
	1	2	3	9	4	10	7	5	6	8
Partition	1	2	3	4	9	10	7	5	6	8

j=4, i=5

Kolik proběhlo kroků - Brentova metoda (Řádek po rekonfiguraci)(5+5b)

odpoved: Nahore

Rekurzivní definice délky lineárního seznamu (4b ?)

odpoved:

Seznam má délku nula pokud je prázdný, jinak má délku jedna plus délka zbytku seznamu

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Shell Sort - přepsat tak, aby se nepřesouvaly hodnoty, ale indexy v poli Por[] (6b ?)

odpoved:

$A[j] > A[j+step] \rightarrow A[Por[j]] > A[Por[j+step]]$

$A[j] := A[j+step] \rightarrow Por[j] := Por[j+step]$

Mnohacestné setřídování - 2000 souborů, 8 pásek, napsat vzorec a zaokrouhlit výsledek (5b)

odpoved:

$2K = 8$

$K = 4$

$N = 2000$

$\log_4 2000$

odhadem $x = 5.5$

Binární vyhledávací strom se zpětnými ukazateli - zadáno 11 hodnot, ty vložit do prázdného stromu a potom odebrat jeden uzel a nakreslit jak po tom bude strom vypadat (3+2b)

odpoved:

viz v předchozích řešeních (tvorba BVS) Smazání proběhne stejně jako u normálního BVS (opět viz výše) a přefigurují se zpětné ukazatele

Max. 30 slovy popsat rušení jednoho prvku v jednosměrném lineárním seznamu bez použití cyklů. (5b)

odpoved:

Je-li seznam jednoprvkový aktivita, první a poslední nastavím na NULL. Mažu-li první uložím si první, prvním se stane druhý a mažu uložený, Mažu-li poslední, ukazatel předposledního (aktuálního) se ruší. Jinak ukládám ukazatel mazaného, mažu prvek a do ukazatele aktivního je načten uložený uk.

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Doplnit kód u nerekurzivního QuickSortu s optimalizovaným zásobníkem (?b)

odpoved:

```
procedure NonRecQuicksort (left,right:integer);
var
  i,j:integer;
  S:TStack; (* deklarace ADT zásobník *)
begin
  SInit(S); (* inicializace zásobníku *)
  Push(S,left); (* vložení levého ind, prvního segmentu *)
  Push(S,right); (* vložení pravého ind. prvního segmentu *)
  while not Empty do
    begin (* vnější cyklus *)
      Top(S,right);
      Pop(S); (* čtení ze zásobníku – reverzace pořadí *)
      Top(S,left);
      Pop(S);
      while left<right do
        begin (* vnitř. cyklus–dokud je co dělit *)
          Partition(A,left,right,i,j);
          -- Push(S,i); (* uložení intervalu pravé části do zásobníku *)
          -- Push(S,right);
          -- right:=j; (* příprava pravého indexu pro další cyklus *)
          ++ if((Right - i) > (j - Left)) then
            ++ begin
              ++   Push(S, i);
              ++   Push(S, Right);
              ++   Right := j;
            ++ end
          ++ else
            ++ begin
              ++   Push(S, Left);
              ++   Push(S, j);
              ++   Left := i;
            ++ end
          end; (* while *)
        end (* while *)
      end; (* procedure *)
```

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

Skupina B:

List merge sort doplnit tabulku + jaká struktura se použije (fronta nebo zásobník) (6+4b)

odpoved:

je použita fronta. Stabilita se musí zajistit tím, že se použije obostranně ukončená fronta (DEQUE)

Ind	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	2	5	9	3	7	10	16	5	8	6	3	11	18	20
Pom	2	3	0	5	6	7	0	9	0	0	12	13	14	0

TRP - 2HASH (na to otázka kolik proběhlo kroků) a Brentova metoda (Řádek po rekonfiguraci) (10b)

odpoved: nahore

Definice ADT (4b)

odpoved:

je definovaný množinou hodnot, kterých smí nabývat každý prvek tohoto typu a množinou operací nad tímto typem

Partition - přepsat tak, aby se nepřesouvaly hodnoty, ale indexy v poli Por[] (6b)

odpoved: nahore

Mnohacestné setřídování - 2000 souborů, 16 pásek, napsat vzorec a zaokrouhlit výsledek (5b)

odpoved: nahore

odhadom $x = 3.5$

Binární vyhledávací stromy se zpětnými ukazateli - zadáno 11 hodnot, ty vložit do prázdného

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

stromu a potom odebrat jeden uzel a nakreslit jak po tom bude strom
vypadat (3+2b)

odpoved: nahore

Max. 30 slovy popsat řešení vložení prvku v jednosměrném lineárním
seznamu před aktivní (5b)

odpoved:

Vložím nový prvek metódou postInsert za aktivní prvek, vezmem hodnotu aktivního prvku a
zapišu ju do nového prvku, hodnotu aktivního prvku prepíšem novými dátami a posuniem
aktivitu (succ)

Doplnit kód u QuickSortu s optimalizovaným zásobníkem (6b)

odpoved: nahore

Řádný termín 2013

Skupina B:

Upravit zadaný kód v MaxScalu shell sortu tak, aby nepřesouval prvky
v poli, ale pouze indexy v pomocném poli.

odpoved: nahore

Quick Sort na datech - jaký bude stav pole po algoritmu Partition + kde
skončí pomocné indexy. (5 + 3b)

odpoved: nahore

Doplnit kód pro rekurzivní ekvivalenci struktur dvou binárních
stromů.

odpoved: nahore

Brentova varianta + TRP na datech + určit, kolik kroků udělá TRP, než
vloží prvek.

odpoved: nahore

d'ování - máme 8 pásek a 1000 souborů, kolik je fází?

odpoved: nahore

odhadom $x = 4.9$

Rekurzivní definice ekvivalence dvou seznamů. (4b)

odpoved: nahore

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

Čím se liší BS se zpětnými ukazateli od normálního BS a implementace jakých operací se liší od implementace operací v normálním BS. (4b)

odpoved:nahore

Opravný termín 2013

1. listmerge sort - ukazatele, stav fronty začátků (6+2b)

odpoved:nahore

2. převést UDPP na DPP (6b)

odpoved:???

Ptr je ukazatel PtrI je index

nil => NilDMA (* const NilDMA=0 *)

Ptr => PtrI

Ptr^ => ArrItem[PtrI]

Ptr^.RPtr => ArrItem[PtrI].RPtr

Ptr^.RPtr^.LPtr => ArrItem[ArrItem[PtrI].RPtr].LPtr

poznámka: nema to byt naopak? to co je tam ted napsane je prevod DPP na UDPP.

ze skript:

Zápis s ukazatelem - DPP	Zápis s indexem - UDPP
Ptr je ukazatel	PtrI je index
nil	NilDMA (const NilDMA=0)
Ptr	PtrI
Ptr^	DMA.ArrItem[PtrI]
Ptr^.RPtr	DMA.ArrItem[PtrI].RPtr
Ptr^.RPtr^.LPtr	DMA.ArrItem[DMA.ArrItem[PtrI].RPtr].LPtr

3. ordery stromu jenž je degradovaný na seznam - pre, in, post, invpre (2+2+2+2b)

odpoved:

4. kod postorder BS do dvousměrného seznamu (6b)

odpoved:

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

5. dijskra na max 10 slov - funkce + použití (3b)

odpoved:

pouziva sa na binárne vyhľadavanie

-vychádza z predpokladu že moze byt viac položiek s rovnakym klucom

-zabezpečuje stabilitu pri vkladani s bin. vyhľadávaním

6. preinsert na 1 směr. seznam na max 30 slov (6b)

odpoved:hore..

7. doplnit queInIt, queUp, queFull (6b)

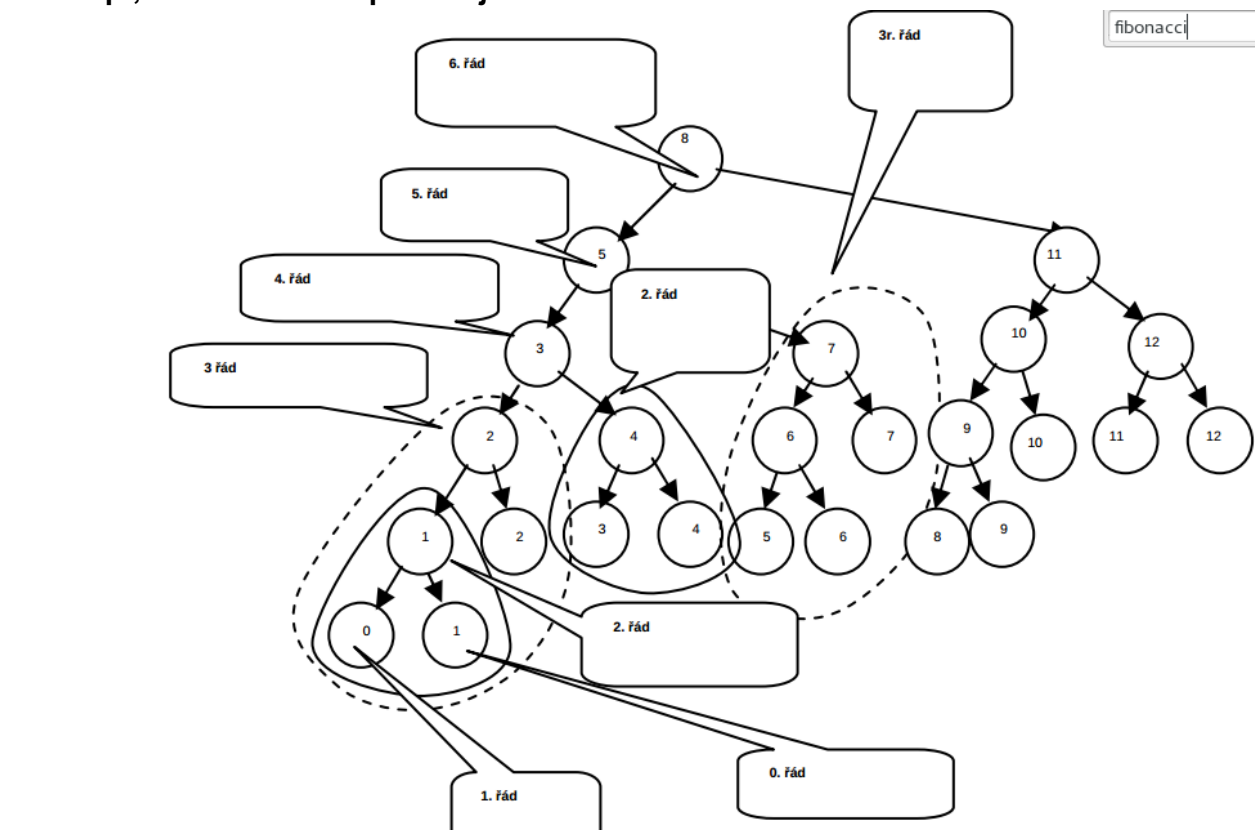
8. mazání postupně tří uzlů v BS (6b)

odpoved:

9. namalovat fibonačiho strom 5. řádu (5b)

odpoved:

zo skript, nakreslis kolko potrebuješ radov



Zadanie riadny 2012:

LEGENDA:

ODPOVED: nevyplnené, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

Skupina A:

- 1 zadane pole hodnot, doplnit partition radek a hodnoty indexu

odpoved:hore

- 2 doplnit 3 iterace shaker sortu

odpoved:??????

- 3 definice ADT

odpoved:nahore

- 4 mazani prvku v jednosmernem seznamu (30 slov max)

odpoved:nahore

- MacLarenův něco, popsat jak funguje, a nahodit první iterace či co, max 8 radku

- definice vyrazu

odpoved:

MacLarenův algoritmus uspořádá pole seřazené bez přesunu na **místě samém** (lat. in situ) - tedy proces bez pomocného pole.

kod:

```
i:=1; Pom:=Prvni;  
while i<Max do begin  
  (* Hledání následníka přesunutého na pozici větší než i *)  
  while Pom<i do Pom:=Pole[Pom].Uk;  
  (* výměna akt. prvního s akt. minimálním *)  
  Pole[i] := Pole[Pom];  
  Pole[i].Uk := Pom; (* stejná výměna ukazatelů *)  
  i:=i+1 (* prvních i-1 prvků je již na svém místě *)  
end;
```

Pozn. Komentář k MacLarenovu algoritmu.

První prvek seznamu (na který ukazuje proměnná Prvni) se vymění s prvkem pole na indexu 1. Tím se nejmenší položka dostane na své místo. Na prvek, který byl z prvního indexu pole odsunut jinam však některý prvek ukazoval. Je třeba ho najít a změnit jeho ukazatel tak, aby místo na první index ukazoval na místo, kam byl první odsunut.

Tím je první prvek ošetřen. Dalším „prvním“ se stane index o jednu větším a cyklus pokračuje tak dlouho, až se vymění předposlední (Max-1) prvek, kdy cyklus končí.

S ohledem na velkou délku položky je součet času řadicího algoritmu bez přesunu položek (minimálně lineární) s časem McLarenova algoritmu (lineární) kratší, než čas samotného řadicího algoritmu s přesunem položek.

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

-5 zkontrolovat maxskvil na mazani v obousmernem seznamu

odpoved:

- 6 upravit maxskvil sift funkci aby byla stable

odpoved:

Sift-down,
rekonfigurace
heapu

Při implementaci binárního stromu polem je důležité poznat konec větve (terminální uzel, nebo uzel který nemá pravého syna):

- Je-li dvojnásobek indexu uzlu větší než počet prvků pole N, pak odpovídající uzel je terminální.
- Je-li dvojnásobek indexu uzlu roven počtu prvků N, má odpovídající uzel pouze levého syna.
- Je-li dvojnásobek indexu uzlu menší než počet prvků N, má odpovídající uzel oba syny.

```
procedure SiftDown(var A:TArr;Left,Right:integer);
(* Left je kořenový uzel porušující pravidla heapu, Right je velikost pole *)
var
  i,j:integer;
  Cont:Boolean; (* Řídící proměnná cyklu *)
  Temp:integer; (* Pomocná proměnná téhož typu jako položka pole *)
begin
  i:=Left;
  j:=2*i; (* Index levého syna *)
  Temp:=A[i];
  Cont:=j<=Right;
  while Cont do begin
    if j<Right
    then (* Uzel má oba synovské uzly *)
      if A[j]< A[j+1]
      then (* Pravý syn je větší *)150
        j:=j+1; (* nastav jako většího z dvojice synů *)
      if Temp >= A[j]
      then (* Prvek Temp již byl posunut na své místo; cyklus končí *)
        Cont:=false
      else begin (* Temp propadá níž, A[j] vyplouvá o úroveň výš *)
        A[i]:=A[j]; (* *)
        i:=j; (* syn se stane otcem pro příští cyklus**)
        j:=2*i; (* příští levý syn *)
        Cont:=j<=Right; (* podmínka : "cyklus pokračuje" *)
      end (* if *)
    end; (* while *)
  A[i]:=Temp; (* konečné umístění prosetého uzlu *)
end; (* procedure *)
```

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

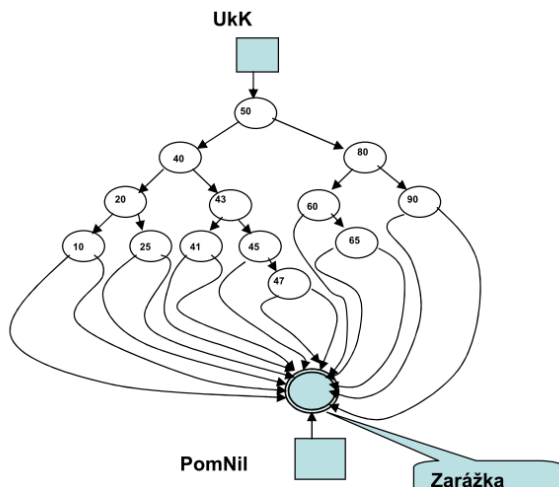
ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

- 7 zakreslit do grafu stromu tak aby byl strom se zarazkou

odpoved:

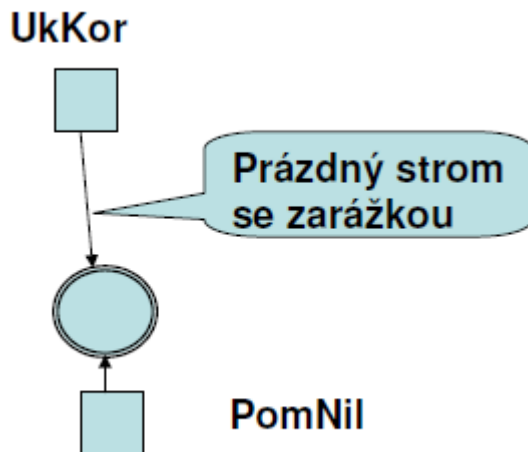
Když už bude nějaký strom a bude třeba dokreslit zarážku, tak prostě každý volný ukazatel uzlů nasměruješ na zarážku.



- 9 nakreslit jak bude vypadat strom se zarazkou po operaci init

odpoved:

po operaci INIT bude strom vyzerat takto:



- 10 popsat rozdíl mezi BVS a BVS se zpětnými ukazateli

odpoved:

- 11 nějaké další 3 otázky k BVS se zpětnými ukazateli

odpoved:

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

- 12 nejakej silenej vyraz prepsat do postfix notace

odpoved:nahore

- 13 napsat fail vektor ke KMP

odpoved:nahore

- 14 Fibonacciho posloupnost 3. radu

odpoved:

*neviem ako to bolo v zadaní ale asi by tam mal byt zadaný nejaký cieľ pokiaľ sa to má počítat.
tak napr. po 8.člen by to bolo ...*

index	0	1	2	3	4	5	6	7	8
	0	0	0	1	1	2	4	8	15

inicializacia pre 3rad .. potom dalsie dopocitavame po 4roch

Skupina B:

1) Vytvoření heapu, shaker sort (5+3b)

odpoved:

2) Vektor fail (6b)

odpoved:nahore

3) Definice binárního stromu (2b)

odpoved:

BS je buď prázdný alebo pozostáva z 1 uzlu (koreňa) a 2 binarnych podstromov (Lavého a Pravého), oba tieto podstromy majú vlastnosti stromu.

Binární strom je buď prázdný, nebo sestává z jednoho uzlu zvaného kořen a dvou binárních podstromů - levého a pravého. (Oba podstromy mají vlastnosti stromu).

LEGENDA:

ODPOVED: nevyplnené, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

4) Fibonacciho posloupnost 2. řádu (4b)

odpoved:

neviem ako to bolo v zadaní ale asi by tam mal byť zadaný nejaký cieľ pokiaľ sa to má počítat. tak napr. po 8.člen by to bolo ...

index	0	1	2	3	4	5	6	7	8
	0	0	1	1	2	4	7	13	24

inicializacia pre 2rad .. potom dalsie dopocitavame po 3roch

5) Popsat vkládání prvku před prvek v jednosměrném seznamu (4b)

odpoved: nahore

6) Popsat Sharovu metodu (3b)

odpoved:

Sharova metoda řeší případ, kdy skutečná velikost (počet prvků) tabulky je jiná, než je hodnota vhodná pro Uniformní binární nebo Fibonacciho vyhledávání. Metoda postupuje ve dvou krocích:

- V prvním kroku provede rozdělení na největším indexu, který vyhovuje metodě a který je menší než daná velikost.
- Ve druhém kroku zjišťuje, kde je hledaný klíč nalevo nebo napravo od dělicí hodnoty. Když je nalevo, postupuje jako by tabulka měla počet prvků daný rozdělovací (a pro metodu vhodnou) hodnotou. Když je napravo, provede transformaci tabulky posunem začátku pole doprava tak, aby prohledávaná část tabulky měla opět vyhovující počet prvků.

7) Rozdíl mezi 1. a dalšími průchody při radix sortu (3b)

odpoved:

V prvom cykle prechadza pole podľa indexu , v ostatných podľa ukazatel

8) Přepsat partition na partition bez přesunu položek (6b)

odpoved:

9) vyčíslení postfixového výrazu (3b)

odpoved:

10) Opravení chyb - rušení 1. prvku v dvousměrném seznamu (6b)

odpoved:

LEGENDA:

ODPOVED: nevyplnené, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

11) Popsat, jak se z heapu dělá pole, podmínky, že je prvek terminální, jen s levým synem (3b)

odpoved:

nie som si istý:

1. najprv musí byť heap "preublany"
2. zoberieme najspodnejši-najpravejší a zameníme ho z korenom..
3. koren môžeme dať do pola
4. zasa "preublame" a pokračuje 2. krokom

preublanie - znamená, že všetci otcovia majú pod sebou menších synov, robí sa to postupne od najspodnejšieho-najpravejšieho.

12) Dokreslení zpětných ukazatelů (3b)

odpoved:

1.Opravy 2012:

1.Merge sort - 3 houpací cykly

odpoved:

2.převod shell sort algoritmu aby se nepřesouvali položky

odpoved:

3.opravit chyby function Search

odpoved:

4.rekurzivní definice seznam

odpoved:

5.pole heap udelat průchod preorder, inorder a postorder

odpoved:

6.jaka je potřeba minimální velikost zásobníku pro 2000 prvků

odpoved:

7.co dělá stabilním ListMerge sort

odpoved:

Stabilita se musí zajistit tím, že se použije obostranně ukončená fronta (DEQUE) do níž se uloží začátky seznamů.

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete kontrolovat' :)

8.opravit chyby v kode PostOrder

odpoved:

```
procedure PostOrder (UkTree:TUk; var List:TList);
var
  Zleva : Boolean;
begin
  SInitBool; (* inicializace zásobníku booleovských hodnot *)
  SInitUk; (* inicializace zásobníku ukazatelů *)
  InitList(List); 250
7.11.2014 , Verze 14-Q
  InsertFirst(List,0); (* vytvoření hlavičky *)
  First(List); (* první je aktivní *)
  Nejlev(UkTree);
  while not SEmptyUk do begin
    TopBool(Zleva); PopBool;
    TopUk(UkTree);
    if Zleva then begin
      PushBool(false); (* vložení příznaku "příště přijde zprava" *)
      Nejlev(UkTree^.PUk);
    end else begin
      PopUk;
      PostInsert(List, UkTree^.Data); (* postupné vkládání do seznamu *)
      SuccList(List); (* postup aktivity *)
    end; (* if *)
  end; (* while *)
  DeleteFirst(List); (* zrušení nepotřebné hlavičky *)
end; (* procedure *)
```

9.k čemu se pouziva Dijskrt

odpoved:

pouziva sa na binárne vyhľadavanie

-vychádza z predpokladu že v moze byt viac poloziek s rovnakym klucom

10.Brentova varinta popsat kde se pouziva

odpoved:

-je to varianta metody TRP s dvomi rozptylovacími funkciami.

-je vhodna za podmienky, keď počet prípadov úspešného vyhľadávania je častejší , než neúspešného vyhľadávania s nasledným vkladáním

11.AVL strom rotace LL

odpoved:

12.kod na optimalizaci zasobniku Quicksortu

odpoved: hore

LEGENDA:

ODPOVED: nevyplnené, treba doplniť

ODPOVED: nejasnosti, treba overiť

ODPOVED: vypracované, ale môžete skontrolovať :)

2.opravný 2012

-1.bubble insert sort (vykonat par cyklov)

odpoved:

- quick sort (vykonat par cyklov)

odpoved:

- 2.asymptotická zložitost (neviem presne čo, nevedel som to)

odpoved:

- 3.ekvivalencia dvoch zoznamov (poucka)

odpoved:

Dva zoznamy su ekvivalentne ak su oba prazdne, alebo sa rovnaju ich prve prvky a sucasne ich zbytky.

- 4.DeleteFirst (doplnit/opravit kod)

odpoved:

DoubleLinked:

```
(*>>>>>>>>>> DDDeleteFirst >>>>>>>>>>*)
```

```
procedure DDeleteFirst
```

```
(var DList:TDList);
```

(* Zruseni prvniho prvku *) 231

7.11.2014 , Verze 14-Q

var

DPomUk:TDUk;

begin

with DList do

begin

```
if Zac<>nil (* je prazdny ? *)
```

then begin

DPomUk:=Zac;

if Zac=Kon (* obsahuje jediny ? *)

then begin

Zac:=nil;

Kon:=nil;

Act:=nil;

```
MarkUsable:=false;
```

end else begin

if Zac=Act (* rusi se aktivni?*)

```
then Act:=nil;
```

LEGENDA:

ODPOVED: nevyplnené, treba doplniť

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

```

if Zac=Mark
then MarkUsable:=false;
Zac:= DPomUk^.Puk;
Zac^.LUk:=nil;
end; (* if Zac= *)
dispose(DPomUk);
end; (* if Zac<> *)
end; (* with *)
end; (* procedure *)

```

SingleLinked:

(* >>>>>>> DeleteFirst <<<<<<<<<<<<<<< *)

```

(* Zrus prvni prvek *)
procedure DeleteFirst(var L:TList);
var
  PomUk:TUk;
begin
  if L.Zac<>nil
  then begin
    PomUk:=L.Zac;
    if L.Zac=L.Act
    then L.Act:=nil;
    L.Zac:=L.Zac^.Uk;
    if L.Zac=nil
    then L.Kon:=nil;
    if PomUk=L.Mark
    then begin
      L.MarkUsable:=false;
      L.Mark:=nil;
    end; (* if *)
    dispose(PomUk);
  end (* if *)
end; (* procedure *)

```

- 5.BVS so spatnymi pointermi (dany kod nejakeho prechodu, bolo treba doplnit chybajuce casti a vylustit aky prechod sa tym algoritmom vykona)

odpoved:

Niekde som cital ze BVS so spatnymi pointermi ma zmysel robit len pri prechode INORDER ked nechceme pouzit rekurziu alebo frontu

- 6.brentova varianta, metoda ci co to je.. (dana tabulka a bolo treba vlozit prvok s nejakym konkretnym klucom)

odpoved: nahore

- 7.ad. vyskove/vahove vyvazeny strom definice.

odpoved:

LEGENDA:

ODPOVED: nevyplnené, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)

BS - je **váňovo** vyvážený, keď pre všetky jeho uzly platí, že počet uzlov Lavého podstromu a Praveho podstromu sa rovnajú alebo líšia o 1.

BS - je **výškovo** vyvážený, keď pre je všetky jeho uzly platí, že výška Lavého podstromu a Pravého podstromu sa rovná alebo sa líši o 1.

- 8. opraviť kód aby prochádzal postorder

odpoved:

```
begin
if RootPtr <> nil
then begin
    Postorder (L, RootPtr^.LPtr) ;
    Postorder (L, RootPtr^.RPtr) ;
    INSERTLAST (L, RootPtr^.Data) ;
end;
end;
```

- 9. něco s 8 páskama, tuším otázka na nějaké dvojice

odpoved:

Další možné zadání

Partition Quick sort - provést a napsat konečné indexy i a j
Insert-Bubble sort - provést 3 iterace
Brentova varianta TRP - vložit klíč do tabulky a vypsát výsledné pole klíčů (pozor to pole bylo indexováno od nuly!)

Pak byl dán kód pro rekurzivní zápis průchodů stromem, mělo se to přepsat tuším na PostOrder. (Šlo myslím jen o to popřehazovat řádky)

Definice výškově a váhově vyváženého stromu.

Pak tam byl aglomerovaný klíč - vytvořit dle zadaného setřídění a byli k tomu funkce na převod int to string.

Napsat nejhorší asymptotickou složitost rozptýlení souboru bo co. (tady si fakt nejsem jistý jak to bylo)

Je dáno tuším že 8 souborů a kolik houpákových cyklů je potřeba u mnohacestného setřídování na setřídění (zde prosím též někoho o upřesnění).

Rekurzivní definice ekvivalence dvou seznamů.

Co je třeba pro rozšíření jednosměrného seznamu na kruhový.

Byla zadána funkce pro průchod stromem se zpětnými ukazateli, mělo se určit o jakou funkci jde a doplnit ji. (Byl to tuším InOrder.)

Doplnit kód pro DeleteFirst nad obousměrným seznamem. (Tady Max blafoval dle mě, bo tam byli dvě okýnka pro doplnění a spodní mělo zůstat prázdné.)

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

- definice ADT
- napsat ve čtyřech bodech o BVS se zpětnými ukazateli
- Quick sort, provést partions nad polem a zapsat výsledné pole
- Bubble insert sort provést 4 cykly a zapsat výsledné pole po každém cyklu
- Popsat jak probíhá mazání u jednosměrného seznamu
- Brentova varianta TRP, měl si zadanou hash tabulku (pole) do které se měl vložit prvek, měl si zapsat výsledné pole po vložení prvku
- Definice výškově vyváženého stromu
- Přepsat část programu, kde se pracovalo s ukazateli na program který pracuje s UDDP (to je to uživatelem definované dynamické přidělování paměti)
- Přepsat část řadičského programu tak, aby pracoval bez přesunu položek - na to se použilo pomocné pole v kterém byly indexy a ty si přesouval
- doplnit u fronty implementované polem 2 nějaké funkce
- doplnit u zásobníku implementovaného seznamem 2 nějaké funkce
- rekurzivní definice ekvivalence dvou seznamů

-
- 1) jak bude vypadat pole čísel po prvním průchodu heap sort (vytvoření hromady, kam se dává největší prvek)
 - 2) vypsát 3 iterace bubble shake sortu, nebo co to bylo, nad zadaným polem čísel z 1)
 - 3) TRP s dvojí rozptylovací funkcí
 - 4) doplnění kódu do 3 funkcí, kde byly vyznačené rámečky na doplnění. (1. bonus: v kódu nemuselo chybět nic. 2. bonus: za doplněný špatný nebo nedoplněný dobrý je -1b dolů, za správné doplnění / vynechání +1b. celkem 7 bodů.)
 - 5) kdy je strom váhově vyvážený? rekurzivní definice (3b)
 - 6) co to je výraz? (2b)
 - 7) nevím, asi definice binárního stromu (2-3b)
 - 8) převod algoritmu bubble-sortu tak, aby se neměnily hodnoty v poli[1..max] (7b, řádek špatně -4b) - měli jsme pro to použít int pole Poradi[1..max]
 - 9) převod zápisu kódu s ukazateli na kód bez ukazatelů (zase +1b/-1b, celkem 8b)
 - 10) podmínky / postup ve třech bodech pro určení lexikografického porovnání dvou řetězců (1. je menší, než 2.)
- a na něco jsem určitě zapoměl.

LEGENDA:

ODPOVED: nevyplněné, třeba doplnit'

ODPOVED: nejasnosti, třeba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

1. Merge Sort

Bylo pole o 20 prvcích, z toho v prvních 10 indexů bylo zaplněno prvky pole.
Úkolem bylo udělat 2 průchody houpačkového cyklu Merge sortu.

2. DeleteFirst – dvousměrný seznam – doplnit kód

Opora - strana 54

3. Kopie seznamu – doplnit kód (2 řádky za begin, podmínka while a tělo while)

Kód by mohl být snad takto:

```
procedure Copy (DLOrig:TDLList; DLDulp:TDLList);
var
```

```
    El: TDtata;
```

```
begin
```

```
    First(DLOrig);
```

```
    InitList(DLDulp);
```

```
    while Active(DLOrig) then begin
```

```
        Copy(DLOrig, El);
```

```
        InbserLast(DLDulp, El);
```

```
        Succ(DLOrig);
```

```
    end;
```

```
end;
```

4. asymptotická složitost rozptylu (? Nepamatuju přesně otázku)

5. napsat vztah nějakýho P, C .. cosi, fakt nevím..

6. příklad s klíčem - nestihla jsem, cosi převést na integer tuším.

7. Upravit kód: vyhledávání v neseřazeném poli upravit na vyhledávání s adaptabilní rekonfigurací

kód byl ze strany 92

Když se prvek najde, tak by se měl prohodit s levým sousedem – cíl: položky, které jsou nejčastěji vyhledávány by měly být na začátku pole.

Kód s úpravou (snad je to Ok):

```
function Search(T:Tab; K:TKlic):Boolean;
```

```
(* funkce vrací hodnotu "true" v případě nalezení prvku s hledaným klíčem *)
```

```
var
```

```
    i:integer; Nasel:Boolean;
```

```
begin
```

```
    Nasel:=false;
```

```
    with T do begin
```

```
        i:=1;
```

```
        while not Nasel and (i<=N) do begin
```

```
            if K = Tab[i].Klic
```

```
            then begin
```

```
                Nasel:=true;
```

```
                if i > 1
```

```
                then
```

```
                    Tab[i]:=Tab[i-1];
```

```
            end else begin
```

```
                i:=i +1
```

```
            end;
```

```
        end; (* while *)
```

```
        Search:=Nasel
```

```
    end (* with *)
```

```
end; (* function*)
```

8. Fibonacciho strom 5. řádu - nakreslit

Opora – strana 100

9. Slovně popsat ve 4 krocích vyčíslování výrazu v postfixové notaci, máme operandy, dyadické operátory a ukončovací znak rovnítko.

Opora – strana 60

10. Diagram signatury jednosměrného seznamu

Opora – strana 43

1. Vytvorit heap, zapsat pozice lichych, vyslo mi to tusim stejne jako vam(5b.)
2. ListMergeSort - doplnit radek Link, zapsat pozice nul (2,5b.)
3. ListMergeSort - doplnit radek Data, zapsat pozice lichych (2,5b.)
4.

0	1	2	3	4	5	6
22	38	61				

doplnit 15 nejakym dvojitym hashem, mi se libila pozice 2, co je spravne nevim (5b.)
5. I. InsertFirst(L2, Pom); (2,5b. ?)
6. J. Nic (2,5b. ?)
 Zadani 5,6: Doplnite kod procedury Invert ktera vyvotri seznam s opacnym poradim prvku

```

Init (L2);
First(L1);
While Active(L1) Do begin
  Copy(L1, Pom);
  --- otazka 5. ---
  Succ(L1);
  --- otazka 6. ---
end;
```
7. H. $Ptr.^{RPtr}LPtr := Ptr.^LPtr$; (5b.)
8. A. $Ptr.^LPtr.^RPtr := Ptr.^RPtr$; (5b.)
 Zadani: Operace Delete nad obousmernym seznamem, chybi radek v podmince kdyz je mazany prvek zaroven první, chybi radek v kody kde je mazany prvek uprostred.
- 9,10: (10*2b.)
 A. Heapsort - je operace Sift linearne slozita? je razeni stabilni?
 B. ---
 C. Quicksort - kdyz davame na zasobnik indexy vetsi casti a radime tu mensi, usporime misto na zasobniku?
 D. Shellsort - neco s bublinovym razenim a NESnizujicim se prirustkem.
 E. Algoritmus KMP - kdyz se automat dostane do stavu Konec jeste pred skoncenim prohledavaneho textu znamena to uspesne hledani?
 F. - J. Dal nevím, jeste tam bylo jestli tento kod: $x=Chr(Ord(x)+Ord(y)); y=Chr(Ord(x)-Ord(y)); x=Chr(Ord(x)-Ord(y));$ prohodí znaky x, y.

1-3. Průchody na stromu po zrušení horního kořene

4. Quick sort

5. Merge sort

6. Fibonacci strom čtvrtého řádu, průchod preorder

7.

Algoritmus z hlavy - používám pseudo C. Snad v tom přepisu nemám chybu.
 Zásobníky S,SB, ukazatel Uk, list L.

LEGENDA:

ODPOVED: nevyplněné, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale můžete skontrolovat' :)

1. Operace nad frontou - pojmenovat doplnit
2. Algoritmus hledání nad BVS se zarážkou - doplnit.
3. Rotace LL - čekal jsem sady ale bylo to úplně jednoduché, jen doplnit druhou půlku obrázku ze skript.
4. Na co je sharova metoda, v cem je lepsi serazene pole
5. Rekurzivní ekvivalence nad lineárním seznam
6. Algoritmus nad obousměrným seznamem, pojmenovat
7. Doplnit algoritmus vyhledávání v poli se zarážkou s adaptivní rekonfigurací prvků
8. Quicksort s optimalizovaným zásobníkem, doplnit
9. Složitost Omikron a Theta, nakreslit graf
10. Složitost heapsortu a proč

-
1. Vytvorit heap, zapsat pozice lichych, vyslo mi to tusim stejne jako vam(5b.)
 2. ListMergeSort - doplnit radek Link, zapsat pozice nul (2,5b.)
 3. ListMergeSort - doplnit radek Data, zapsat pozice lichych (2,5b.)
 4.

0	1	2	3	4	5	6
1	22	38	61			

 doplnit 15 nejakym dvojitym hashem, mi se libila pozice 2, co je spravne nevim (5b.)
 5. I. InsertFirst(L2, Pom); (2,5b. ?)
 6. J. Nic (2,5b. ?)
Zadani 5,6: Doplnite kod procedury Invert ktera vyvotri seznam s opacnym poradim prvků
Init (L2);
First(L1);
While Active(L1) Do begin
Copy(L1, Pom);
--- otazka 5. ----
Succ(L1);
--- otazka 6. ----
end;
 7. H. $Ptr.^{RP}Ptr.^{LP}Ptr := Ptr.^{LP}Ptr$; (5b.)
 8. A. $Ptr.^{LP}Ptr.^{RP}Ptr := Ptr.^{RP}Ptr$; (5b.)
Zadani: Operace Delete nad obousmernym seznamem, chybi radek v podmince kdyz je mazany prvek zaroven prvni, chybi radek v kody kde je mazany prvek uprostred.
 - 9,10: (10*2b.)
A. Heapsort - je operace Sift linearne slozita? je razeni stabilni?
B. ---
C. Quicksort - kdyz davame na zasobnik indexy vetsi casti a radime tu mensi, usporime misto na zasobniku?
D. Shellsort - neco s bublinovym razenim a NESnizujicim se prirustkem.
E. Algoritmus KMP - kdyz se automat dostane do stavu Konec jeste pred skoncenim prohledavaneho textu znamena to

-
- 1/ Mas dva nenilove ukazatele na frontu realizovanou jednosmernym seznamem s DPP. 1. uk ukazuje na zacatek. Co vsechno muzes smazat?
 - 2/Co delas s operandem pri prevodu z infixu do postfixu.
 - 3/Nevsponimam si... EDIT: Mas kod a uhodni co to je. uokzaraz es dohcurs?
 - 4/Fronta od 0..N, Jaka je poloha zacatku po operaci remove?
Odpoved stylu: (Zac + 1) mod (N +1)
 - 5/Co je backtracking?
 - 6/Premie: Prevod hutneho vsorce z infixu do postfixu. Celkem 21 opratoru a operandu.
 - 7/ Dopl chybejici kod v nerekursivnim quicksortu
 - 8/Dopl kod v Prepisu binarniho stromu se zpetnymi ukazateli inordrem do seznamu.

LEGENDA:

ODPOVED: nevyplnené, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale možete skontrolovať :)

Staršie:

1. jako 7. preorder uzel C; odpoved C | 1b
2. jako 7. inorder uzel H; odpoved H | 1b
3. jako 7. postorder uzel I; odpoved I | 1b
4. spocitat prefix, vysledek 5 (odpoved F) | 2b
5. spocitat postfix, vysledek 6 (odpoved G) | 2b
6. 3D mapovaci funkce po radcich | 2b
7. 3D mapovaci funkce po sloupcich | 3b
8. ADT vyhledavaci tabulka signatury nebo tak neco :mrgreen: | 2b

SK.A:

1. Doplnenie funkcie deleteDMA (4b)
2. Doplnenie funkci Queue (Full, Remove, QueUp) (5b)
3. Diagram signatury ADT zoznam (3b)
4. Definovat dlzku zoznamu rekurzivne na max. 25 slov. (2b)

SK.B:

1. Doplneni funkce deleteDMA (4b)
2. Doplneni funkci Queue (Full, Remove, QueUp) - zøetizeným seznamem (5b)
3. Diagram signatury ADT fronta (3b)
4. Definovat ekvivalenci seznamu rekurzivne na max. 25 slov. (2b)p

LEGENDA:

ODPOVED: nevyplnené, treba doplnit'

ODPOVED: nejasnosti, treba overit'

ODPOVED: vypracované, ale môžete skontrolovať :)