Výstupy RAM/RASP:

26. Ktoré z uvedených obsahov vstupnej pásky spôsobia, že na výstupnú pásku stroja RAM po vykonaní uvedeného programu bude zapísaná hodnota 7?

```
READ 2
LOOP: READ 0
JZ FIN
DIV =2
ADD 2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) 1,2,4,3,5,0 //1,5b
b) 2,1,3,5,4,0 //1,5b
c) 0,4,3,2,5,0
d) žiadny z uvedených

63. Ktoré z uvedených obsahov vstupnej pásky spôsobia, že na výstupnú pásku stroja RAM po vykonaní uvedeného programu bude zapísaná hodnota väčšia ako 7?

```
READ 2
LOOP: READ 0
SUB =2
JZ FIN
ADD 2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) 1,4,3,5,2,0
b) 2,1,3,5,2,0
c) 0,4,3,2,5,0
d) žiadny z uvedených //3b

71. Ktoré z daných vstupov spôsobia, že na výstupnú pásku stroja RAM po vykonaní uvedeného programu bude zapísaná hodnota väčšia ako 15?

```
READ 1
LOAD =1
STORE 2
LOOP: LOAD 1
JZ FIN
SUB =1
STORE 1
LOAD 2
MUL =2
STORE 2
JMP LOOP

FIN: WRITE 2
HALT
```

a) 4 //1,5b
b) 2
c) 8 //1,5b
d) 3

104. Ktoré z uvedených obsahov vstupnej pásky spôsobia, že na výstupnú pásku stroja RAM po vykonaní uvedeného programu bude zapísaná hodnota menšia ako 7?

```
READ 2
LOOP: READ 0
JZ FIN
DIV =2
ADD 2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) 1,2,4,3,5,0
b) 2,1,3,5,4,0
c) 0,4,3,2,5,0 //3b
d) žiadny z uvedených

114. Aký výsledok bude zapísaný na výstupnú pásku stroja RAM po vykonaní uvedeného programu, ak vstupná páska bude obsahovať hodnotu 4?

```
READ 1
LOAD =1
STORE 2
LOOP: LOAD 1
JZ FIN
SUB =1
STORE 1
LOAD 2
MUL =2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) 16
b) 8
c) 32 //3b

148. Abstraktný stroj RAM obsahuje tieto súčasti:

a) Pamäť dát //0,75b
b) Pamäť programu //0,75b
c) Vstupná páska //0,75b
d) Výstupná páska //0,75b

159. Abstraktný stroj RASP obsahuje tieto súčasti:

a) Pamäť skokov
b) Pamäť programu
c) Vstupná páska //1,5b
d) výstupná páska //1,5b

150. Aký výsledok bude zapísaný na výstupnú pásku stroja RAM po vykonaní uvedeného programu, ak vstupná páska bude obsahovať potupnosť: 5,2,4,5,1,3,7,0?

```
READ 1
LOAD =0
STORE 2
LOOP: LOAD 1
JZ FIN
SUB =1
STORE 1
READ 0
ADD 2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) 22
b) 12
c) 15//3b
d) 26

171. Aký výsledok bude zapísaný na výstupnú pásku stroja RAM po vykonaní uvedeného programu, ak vstupná páska bude obsahovať potupnosť: 4,2,4,5,1,3,7,0?

```
READ 1
LOAD =0
STORE 2
LOOP: LOAD 1
JZ FIN
SUB =1
STORE 1
READ 0
ADD 2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) 22
b) 12 //3b
c) 15
d) 26

212. Ktoré z uvedených obsahov vstupnej pásky spôsobia, že na výstupnú pásku stroja RAM po vykonaní uvedeného programu bude zapísaná párna hodnota?

```
LOAD =0
STORE 2
LOOP: READ 0
JZ FIN
ADD 2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) žiadny z uvedených
b) 2,4,3,5,0 //1,5b
c) 4,2,0,5,2 //1,5b
d) 3,4,7,8,5

225. Ktoré z daných vstupov spôsobia, že na výstupnú pásku stroja RAM po vykonaní uvedeného programu bude zapísaná hodnota menšia ako 20?

```
READ 1
LOAD =2
STORE 2
LOOP: LOAD 1
JZ FIN
SUB =1
STORE 1
LOAD 2
MUL =2
STORE 2
JMP LOOP
FIN: WRITE 2
HALT
```

a) 2 //1,5b
b) 8
c) 3 //1,5b
d) 4

Ram inštrukcie:

04. Jazyk lineárneho modelu RAM neobsahuje tieto inštrukcie:

a) LOAD
b) ADD
c) HALT //1,5b
d) JMP //1,5b

13. Jazyk modelu bitových výpočtov stroja RAM obsahuje tieto inštrukcie:

a) JMP
b) XOR //1,5b
c) SUB
d) NOT //1,5b

22. Ktorá z nasledujúcich inštrukcií stroja RAM má najvyššiu cenu pri logaritmickom kritériu zložitosti? Nech c(i)=3 pre i<3 a c(i)=4 pre i>2.

a) WRITE *2
b) MUL *2 //3b
c) SUB 2
d) STORE *1

56. Ktorá z nasledujúcich inštrukcií stroja RAM má najvyššiu cenu pri logaritmickom kritériu zložitosti? Nech c(i)=3 pre i<3 a c(i)=4 pre i>2.

a) STORE *2 //3b
b) WRITE 2
c) LOAD =1
d) ADD =1

124. Ktoré z nasledujúcich inštrukcií stroja RAM majú uvedený korektný význam (M<> predstavuje pamäťovú referenciu)?

a) STORE i (r_i <-- r_0) //1,5b
b) LOAD *i (r_i <-- M<r_i>) //1,5b
c) SUB *i (r_o <-- r_0 - r_i)
d) READ i (r_0 <-- vstup)

126. Ktoré z nasledujúcich inštrukcií stroja RAM majú uvedený korektný význam (M<> predstavuje pamäťovú referenciu)?

a) WRITE i (výstup <-- r_i) //1,5b
b) LOAD =i (r_0 <-- r_i)
c) JMP l (r_o <-- l)
d) ADD *i (r_0 <-- r_0 + M<r_i>) //1,5b

164. Aké sú prístupné typy operandov inštrukcie READ stroja RAM?

a) *i //1,5b
b) =i
c) i //1,5b

191. ktoré z uvedených inštrukcií predstavujú korektné inštrukcie stroja RAM?

a) READ =2
b) ADD *5 //1,5b
c) OUT 4
d) HALT //1,5b

252. Ktoré z uvedených inštrukcií predstavujú korektné inštrukcie stroja RAM?

a) STORE =3
b) DIV 5 //1,5b
c) RET
d) WRITE =2 //1,5b

Kódy a programovanie:

14. Zásobníkový rámec pri volaní procedúr neobsahuje:

a) Meno volajúcej procedúry //1,5b
b) Priestor pre lokálne premenné
c) Aktuálne parametre
d) Adresa začiatku volajúcej procedúry //1,5b

31. Ktoré z uvedených čísel sú Fibonacciho čísla (prvého rádu)?

a) 55 //1,5b
b) 34 //1,5b
c) 22
d) 35

49. Pre metódu Divide-and-conquer je charakteristické:

a) Postup zhora-nadol (od riešenia problému k elementárnym podproblémom) //1,5b
b) Postup zdola-nahor (od elementárnych podproblémov k riešenému problému)
c) Použitie rekurzie //1,5b
d) Použitie iterácie
e) Vyhradené použitie aritmetickej operácie celočíselného delenia

68. Ktoré z metód návrhu algoritmov sú využité v prípade uvedeného programu?

```
int Fib(int n){
    int fib[n+1];
    int i;
    fib[0]=1;
    fib[1]=1;
    for(i=2;i<=n;i++) fib[i]=fib[i-1]+fib[i-2];
    return fib[n];
}
```

a) Rekurzia
b) Dynamické programovanie //3b
c) Greedy
d) Vyvažovanie

98. Ktoré z uvedených príkazov priradenia je potrebné doplniť na vyznačenom mieste [?] procedúry MERGE?

```
procedure MERGE(S1,S2,S3):
begin
   nech S1 = {a1,a2,...,an1}, S2 = {b1,b2,...,bn2};
   i ← 1; j ← 1; k ← 1;
   while i ≤ |S1| and j ≤ |S2| do
      begin
         while S1[i] ≤ S2[j] and i ≤ |S1| do
            begin
               S3[k] ← S1[i]; i ← i + 1; k ← k + 1;
            end
         while S2[j] ≤ S1[i] and j ≤ |S2| do
            begin
               S3[k] ← S2[j]; j ← j + 1; k ← k + 1;
            end
      end
   while i ≤ |S1| do
      begin
         S3[k] ← S1[i]; i ← i + 1; k ← k + 1;
      end
   while j ≤ |S2| do
      begin
         [    ?    ];
      end
   return S3;
end
```

a) k <-- k + 1; //1b
b) S3[k] <-- S2[j]; //1b
c) j <-- j + 1; //1b
d) S3[k] <-- S1[i];
e) i <-- i + 1;
f) k <-- k - 1;

117. Ktoré z metód návrhu algoritmov sú využité v prípade uvedeného programu?

```
int Fib(int n){
    if(n==0 || n==1)return 1;
        else return Fib(n-1)+Fib(n-2);
}
```

a) Greedy
b) Dynamické programovanie
c) Rekurzia //1,5b
d) Rozdeľuj a panuj //1,5b

187. Ktoré z metód návrhu algoritmov sú využité v prípade uvedeného programu?

```
int BinKoef(int n,int k){
    int bk[n+1][n+1];
    int i,j;
    bk[0][0]=1;
    for(i=1;i<=n;i++){
        bk[i][0]=1;
        bk[i][i]=1;
        for(j=1;j<i;j++)
            bk[i][j]=bk[i-1][j-1]+bk[i-1][j];
        }
    return bk[n][k];
}
```

a) Rekurzia

b) Greedy

c) Dynamické programovanie //3b

192. Doplňte chýbajúci fragment kódu procedúry MXIM (na obrázku):

```
procedure MXIN(S):
begin
    nech S je reprezentovaná poIom S[i], i=1,2,...,n, n=2k, k ≥ 1
    (max,min) ← (MAX(S[1],S[n]),MIN(S[1],S[n]));
     for([    ?    ])do
            begin
                (MAX,MIN)←(MAX(S[i],S[n-(i-1)]),MIN(S[i],S[n-(i-1)]));
                if(max < MAX) then max ← MAX;
                if(min > MIN) then min ← MIN;
            end
end
```

a) i <-- 2 until i <= n step 1

b) i <-- 1 until i <= n step 1

c) i <-- 1 until i <= n/2 step 1

d) i <-- 2 until i <= n/2 step 1 //3b

196. Technika dynamické programovanie realizuje:

a) najprv riešenie podproblémov menších rozmerov, potom väčších //1,5b

b) najprv riešenie podproblémov väčších rozmerov, potom menších

c) výpočet riešení vybraných podproblémov

d) výpočet riešení všetkých podproblémov //1,5b

226. Jazyk PL využívaný pri prezentácií algoritmov umožňuje používanie:

a) komentárov //0,75b
b) skoku goto //0,75b
c) operácií vstupu a výstupu //0,75b
d) procedúr //0,75b

227. Príkladom využitia ktorej z metód návrhu algoritmov je uvedený kód?

```
int FNC(int n){
    if(n==0)return 0;
        else return FNC(n-1)+n;
}
```

a) Greedy metóda
b) Divide-and-conquer
c) Jednoduchá rekurzia //3b
d) Dynamické programovanie

235. Doplňte chýbajúci riadok kódu procedúry MAXMIN (na obrázku):

```
procedure MAXMIN(S);
begin
    if |S|=2 then
            begin
                mech S={a,b};
                return (MAX(a,b), MIN(a,b))
            end
    else
            begin
                rozdeľ S na S1 a S2, kde |S1|=|S2|=n/2;
                (max1,min1) ← MAXMIN(S1);
                (max2,min2) ← MAXMIN(S2);
                [    ?    ];
            end
end
```

a) return(S2, S1)
b) return(MAX(max1, max2), MIN(min1, min2)) //3b
c) return(S1, S2)
d) return(max1, min1)
e) return(max2, min2)
f) return(MIN(min1, min2), MAX(max1, max2))

247. Pre metódu Dynamické programovanie je charakteristické:

a) Použitie dynamických údajových štruktúr
b) Použitie rekurzie
c) Použitie iterácie //1,5b
d) Postup zhora-nadol (od riešenia problému k elementárnym podproblémom)
e) Postup zdola-nahor (od elementárnych podproblémov k riešenému problému) //1,5b

Prechody:

89. Optimálny binárny vyhľadávací strom je označený stratégiou:

a) PREORDER
b) INORDER //3b
c) LEVELORDER
d) POSTORDER

Postorder:

11. Strom na obrázku je označený stratégiou?

a) Postorder //3b

39. Majme binárny strom reprezentovaný poľom A=(5,7,6,8,9,2,0,0,0,3), kde A[1] je koreň stromu a ľavý potomok A[i] je vždy A[2i], pravý A[2i+1]. Ak A[i]=0, znamená to, že na danej pozícii uzol v strome nie je. Ktorý z nasledujúcich je výpisom uzlov stromu stratégiou postorder?

a) 8, 7, 5, 2, 6, 3, 9
b) 8, 7, 3, 9, 5, 2, 6
c) 8, 3, 9, 7, 2, 6, 5 //3b
d) 5, 7, 8, 6, 2, 9, 3
e) 8, 3, 9, 2, 6, 7, 5
f) 5, 7, 8, 9, 3, 6, 2
g) 8, 3, 2, 6, 9, 7, 5
h) 5, 7, 8, 3, 6, 9, 2
i) 8, 7, 3, 5, 2, 9, 6

86. Pri POSTORDER prechode daným binárnym stromom (na obrázku) budú vypísané hodnoty v poradí:

a) 2,3,4,8,9
b) 9,3,4,2,8
c) 9,3,2,4,8
d) 2,3,8,4,9 //3b

133. Majme binárny strom reprezentovaný poľom A=(5,6,7,1,3,0,8,9,2), kde A[1] je koreň stromu a ľavý potomok A[i] je vždy A[2i], pravý A[2i+1]. Ak A[i]=0, znamená to, že na danej pozícií uzol v strome nie je. Ktorý z nasledujúcich je výpisom uzlov stromu stratégiou postorder?

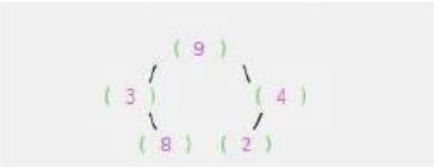
a) 5, 6, 1, 3, 7, 8, 9, 2
b) 5, 1, 6, 9, 3, 3, 2, 8
c) 9, 2, 1, 3, 6, 8, 7, 5 //3b
d) 9, 2, 1, 8, 7, 5, 3, 6
e) 9, 1, 2, 6, 7, 8, 5, 3
f) 5, 6, 1, 9, 2, 3, 7, 8
g) 9, 1, 2, 3, 7, 8, 6, 5
h) 9, 1, 2, 6, 3, 5, 7, 8
i) 6, 5, 1, 9, 2, 3, 8, 7

115. Uvedený kód predstavuje implementáciu prechodu:

```
void TT(int root){
     if(left[root]!=0) TT(left[root]);
     if(middle[root]!=0) TT(middle[root]);
     if(right[root]!=0) TT(right[root]);
     printf("%d ",value[root]);
}
```

a) ternárnym stromom stratégiou Inorder
b) birnárnym stromom stratégiou Preorder
c) birnárnym stromom stratégiou Postorder
d) ternárnym stromom stratégiou Postorder //3b
e) binárnym stromom stratégiou Inorder
f) ternárnym stromom stratégiou Preorder

119. Pri POSTORDER prechode daným binárnym stromom (na obrázku) budú vypísané hodnoty v poradí:



a) 8,3,2,4,9 //3b
b) 2,4,8,3,9
c) 9,3,4,8,2
d) 9,3,8,4,2
e) 8,2,3,4,9

Inorder:

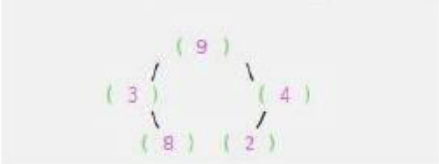
25. Pri použití nerekurzívnej procedúry Inorder pre značenie stromov sa do zásobníka ukladajú:

a) Zásobníkové rámce
b) Vrcholy stromu //3b

44. Pri použití rekurzívnej procedúry Inorder pre značenie stromov sa do zásobníka ukladajú

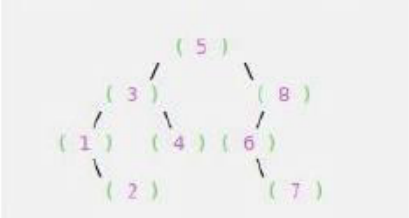
a) Zásobníkové rámce //3b
b) vrcholy stromu

65. Pri INORDER prechode daným binárnym stromom(na obrázku) budú vypísané hodnoty v poradí:



a) 3,8,9,2,4 //3b
b) 8,3,2,4,9
c) 9,3,8,4,2
d) 8,3,9,2,4
e) 3,8,9,4,2

66. Strom na obrázku je označený stratégiou:



a) Preorder
b) Inorder //3b
c) inou stratégiou
d) Postorder

69. Majme binárny strom reprezentovaný poľom A=(5,7,6,8,9,2,0,0,0,3), kde A[1] je koreň stromu a ľavý potomok A[i] je vždy A[2i], pravý A[2i+1]. Ak A[i]=0, znamená to, že na danej pozícií uzol v strome nie je. Ktorý z nasledujúcich je výpisom uzlov stromu stratégiou inorder?

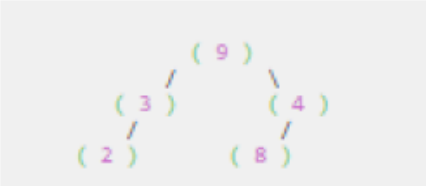
a) 8, 7, 3, 9, 5, 2, 6 //3b
b) 8, 3, 2, 6, 9, 7, 5
c) 5, 7, 8, 6, 2, 9, 3
d) 5, 7, 8, 9, 3, 6, 2
e) 8, 3, 9, 2, 6, 7, 5
f) 8, 7, 3, 5, 2, 9, 6
g) 5, 7, 8, 3, 6, 9, 2
h) 8, 7, 5, 2, 6, 3, 9
i) 8, 3, 9, 7, 2, 6, 5

72. Uvedený kód predstavuje implementáciu prechodu binárnym stromom stratégiou:

```
void NR(int v){
     Stack S;
     S = CreateStack( 12 );
LT:  while(left[v]!=0){
           Push(v,S);
          v=left[v];
         }
NODE:printf("%d ",value[v]);
     if(right[v]!=0){
          v=right[v];
          goto LT;
          }
     if(!IsEmpty(S)){
          v=Top(S);
          Pop(S);
          goto NODE;
     }
     DisposeStack( S );
}
```

a) Levelorder
b) Postorder
c) Preorder
d) Inorder //3b

94. Pri INORDER prechode daným binárnym stromom(na obrázku) budú vypísané hodnoty v poradí:



a) 2,3,9,8,4 //3b
b) 3,2,9,4,8
c) 9,3,4,2,8
d) 2,3,8,4,9
e) 3,2,4,8,9

151. Majme binárny strom reprezentovaný poľom A=(5,6,7,1,3,0,8,9,2), kde A[1] je koreň stromu a ľavý potomok A[i] je vždy A[2i], pravý A[2i+1]. Ak A[i]=0, znamená to, že na danej pozícií uzol v strome nie je. Ktorý z nasledujúcich je výpisom uzlov stromu stratégiou inorder?

a) 9, 1, 2, 3, 7, 8, 6, 5
b) 5, 1, 6, 9, 3, 3, 2, 8
c) 9, 1, 2, 6, 3, 5, 7, 8 //3b
d) 9, 2, 1, 3, 6, 8, 7, 5
e) 9, 1, 2, 6, 7, 8, 5, 3
f) 5, 6, 1, 9, 2, 3, 7, 8
g) 9, 2, 1, 8, 7, 5, 3, 6
h) 5, 6, 1, 3, 7, 8, 9,2
i) 6, 5, 1, 9, 2, 3, 8, 7

189. Ktoré zo stratégií označovanie (prechádzania) stromov možno aplikovať len na binárne stromy?

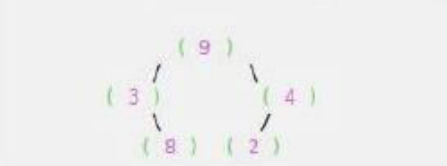
a) Postorder
b) Preorder
c) Level-order
d) Inorder //3b

Preorder:

20. Strom na obrázku je označený stratégiou?

a) Inorder
b) inou stratégiou
c) Preorder //3b
d) Postorder

123. Pri PREORDER prechode daným binárnym stromom (na obrázku) budú vypísané hodnoty v poradí:



a) 9,3,8,4,2 //3b

157. Majme binárny strom reprezentovaný poľom A=(5,7,6,8,9,2,0,0,0,3), kde A[1] je koreň stromu a ľavý potomok A[i] je vždy A[2i], pravý A[2i+1]. Ak A[i]=0, znamená to, že na danej pozícií uzol v strome nie je. Ktorý z nasledujúcich je výpisom uzlov stromu stratégiou preorder?

a) 8, 3, 2, 6, 9, 7, 5
b) 8, 7, 5, 2, 6, 3, 9
c) 5, 7, 8, 9, 3, 6, 2 //3b
d) 8, 3, 9, 2, 6, 7, 5
e) 8, 7, 3, 9, 5, 2, 6
f) 8, 7, 3, 5, 2, 9, 6
g) 8, 3, 9, 7, 2, 6, 5
h) 5, 7, 8, 3, 6, 9, 2
i) 5, 7, 8, 6, 2, 9, 3

207. Uvedený kód predstavuje implementáciu prechodu binárnym stromom stratégiou:

```
void NR(int v){
     Stack S;
     S = CreateStack( 12 );
LT:  printf("%d ",value[v]);
     if(left[v]!=0){
           Push(v,S);
           v=left[v];
           goto LT;
          }
RT:   if(right[v]!=0){
          v=right[v];
          goto LT;
          }
     if(!IsEmpty(S)){
          v=Top(S);
          Pop(S);
          goto RT;
     }
     DisposeStack( S );
}
```

a) Levelorder
b) Postorder
c) Inorder
d) Preorder //3b

257. Majme binárny strom reprezentovaný poľom A=(2,3,4,0,5,6,7,0,0,8,9), kde A[1] je koreň stromu a ľavý potomok uzla A[i] je vždy A[2i], pravý A[2i+1]. Ak A[i]=0, znamená to, že na danej pozícií uzol v strome nie je. Ktorý z nasledujúcich je výpisom uzlov stromu stratégiou preorder?

a) 8, 9, 5, 4, 2, 3, 6, 7
b) 3, 8, 5, 9, 2, 6, 4, 7
c) 2, 3, 5, 8, 9, 4, 6, 7 //3b
d) 2, 3, 5, 6, 7, 8, 9, 4
e) 2, 3, 5, 8, 4, 6, 9, 7
f) 3, 8, 5, 2, 6, 4, 9, 7
g) 8, 9, 5, 3, 4, 6, 7, 2
h) 8, 9, 5, 3, 6, 7, 4, 2
i) 3, 8, 5, 7, 2, 4, 6, 9

Sort:

40. Ktoré z uvedených metód návrhu algoritmov sú využívané algoritmom Merge sort?

a) dekompozícia //1b
b) vyvažovanie //1b
c) dynamické programovanie
d) rekurzia //1b
e) greedy

87. Metóda prirodzeného zlučovania pre zvýšenie efektívnosti vonkajšieho triedenia využíva:

a) starostlivo zvolenú distribúciu počiatočných behov (Fibonacciho čísla)
b) situáciu, ak sú údaje na začiatku čiastočné utriedené //2b
c) distribúciu behov na viac ako dve pásky

127. Procedúra SORT je aplikáciou uvedených metód návrhu algoritmov:

```
procedure SORT(i,j):
begin
    if i=j then return Xi;
        else
            begin
                m ← (i+j-1)/2;
                return MERGE(SORT(i,m),SORT(m+1,j))
            end
end
```

a) Dynamické programovanie
b) Vyvažovanie //1,5b
c) Greedy
d) Rekurzia //1,5b

138. Čau Jožo, ak čítaš tieto riadky,tak som na Teba pyšný ... pretože sa učíš :D

a) Roman
b) Noro
c) Šimoňák //99,7b
d) Feťák
e) Kpt. Andrej Danko //0,3b

239. Medzi triediace algoritmy využívajúce operáciu porovnania triedených prvkov patria:

a) InsertionSort //0,6b
b) MergeSort //0,6b
c) QuickSort //0,6b
d) BubbleSort //0,6b
e) HeapSort //0,6b

245. Použitie metódy Divide-and-conquer je typické pre triediace algoritmy:

a) Heap sort
b) Insertion sort
c) Bubble sort
d) Radix sort
e) Merge sort //3b

Insert:

38. Uvedená postupnosť bola utriedená algoritmom:

2 5 3 8 4 6 7 1 9 (VSTUP)
5 2 3 8 4 6 7 1 9 (l = 2)
5 3 2 8 4 6 7 1 9 (l = 3)
8 5 3 2 4 6 7 1 9 (l = 4)
8 5 4 3 2 6 7 1 9 (l = 5)
8 6 5 4 3 2 7 1 9 (l = 6)
8 7 6 5 4 3 2 1 9 (l = 7)
8 7 6 5 4 3 2 1 9 (l = 8)
9 8 7 6 5 4 3 2 1 (l = 9)

a) Bubble sort
b) Heap sort
c) Radix sort
d) Insert sort //3b

132. Uvedená postupnosť bola utriedená algoritmom:

2 5 3 8 4 6 7 1 9 (VSTUP)
2 5 3 8 4 6 7 1 9 (l = 2)
2 3 5 8 4 6 7 1 9 (l = 3)
2 3 5 8 4 6 7 1 9 (l = 4)
2 3 4 5 8 6 7 1 9 (l = 5)
2 3 4 5 6 8 7 1 9 (l = 6)
2 3 4 5 6 7 8 1 9 (l = 7)
1 2 3 4 5 6 7 8 9 (l = 8)
1 2 3 4 5 6 7 8 9 (l = 9)

a) Bubble sort

b) Heap sort
c) Radix sort
d) Insert sort //3b

Radix:

08. Ktoré z uvedených nádob B[], využívaných algoritmom Radix sort majú uvedený správny obsah, ak j=1 (prostredný znak reťazca) a postupnosť pre utriedenie je: 041 145 169 281 334 358 464 467 478 491 500 705 724 827 961 962?

a) B[5]: 358 145
b) B[6]: 961 962 464 467 169 //1,5b
c) B[4]: 041
d) B[8]: 281 358
e) B[1]: //1,5b

34. Ktoré z uvedených obsahov údajovej štruktúry NONEMPTY využívanej algoritmom Radix sort pre triedenie reťazcov rôznej dĺžky sú korektné pre utriedenie postupnosti reťazcov: cc, a, bc, aab, baca, cbc?

a) NONEMPTY[3] = {b,c,a}
b) NONEMPTY[2] = {c,a}
c) NONEMPTY[1] = {c,a,b} //1,5b
d) NONEMPTY[4] = {a} //1,5b

37. V rámci uvedeného algoritmu triedenia Radix sort, hodnota k predstavuje:

```
begin
   vlož A1, A2, ... , An do QUEUE;
   for j ← k step -1 until 1 do
   begin
      for l ← 0 until m-1 do vyprázdni Q[l];
      while QUEUE ≠ empty do
         begin
            nech Ai je prvý prvok v QUEUE;
            prenes Ai z QUEUE do Q[aij];
         end
      for l ← 0 until m-1 do
         vlož Q[l] na koniec QUEUE;
   end
end
```

a) rozsah hodnôt jednotlivých prvkov (symbolov) reťazca
b) počet triedených reťazcov
c) nič z uvedených možností
d) pozíciu aktuálne spracovaného symbolu v rámci reťazca
e) dĺžku triedených reťazcov //3b

60. Ktoré z uvedených hodnôt predstavujú korektný počet prvkov štruktúry LENGTH využívanej algoritmom Radix sort pre triedenie reťazcov rôznej dĺžky pre utriedenie postupnosti reťazcov: caca, b, cc, a, bc, aab, baca, cbc?

a) |LENGTH[1]| = 2 //1,5b
b) |LENGTH[2]| = 4
c) |LENGTH[3]| = 6
d) |LENGTH[4]| = 2 //1,5b

73. Ktoré z uvedených nádob B[], využívaných algoritmom Radix sort majú uvedený správny obsah, ak j=2 (posledný znak reťazca) a postupnosť pre utriedenie je: 041 145 169 281 334 358 464 467 478 491 500 705 724 827 961 962?

a) B[4]: 334 464
b) B[6]: //1,5b
c) B[7]: 478 827
d) B[8]: 358
e) B[5]: 705 145 //1,5b

74. Algoritmus Radix sort pre triedenie reťazcov rôznej dĺžky využíva pri spájaní obsadených nádob (frontov) údajovú štruktúru:

a) LENGTH
b) TREE
c) STACK
d) NOEMPTY //3b

113. Ktoré z uvedených hodnôt predstavujú korektný počet prvkov štruktúry LENGTH využívanej algoritmom Radix sort pre triedenie reťazcov rôznej dĺžky pre utriedenie postupnosti reťazcov: cc, a, bc, aab, baca, cbc?

a) |LENGTH[3]| = 2 //1,5b
b) |LENGTH[2]| = 5
c) |LENGTH[1]| = 6
d) |LENGTH[4]| = 1 //1,5b

116. Pri triedení algoritmom Radix sort pre triedenie reťazcov rôznej dĺžky po i-tom prechode hlavnou slučkou v QUEUE sú iba reťazce dĺžky:

a) $/_{max}$-i+1 a väčšej //3b
b) $/_{max}$-i a menšej
c) $/_{max}$-i+1 a menšej
d) $/_{max}$-i a väčšej

134. Údajová štruktúra NONEMPTY[i] (zoznam) využívaná v rámci algoritmu Radix sort pre triedenie reťazcov rôznej dĺžky obsahuje:

a) všetky symboly, ktoré sa vyskytujú v i-tom triediacom reťazci
b) všetky symboly, ktoré sa vyskytujú v i-tej pozícií niektorého z triedených reťazcov //3b
c) žiadna z uvedených možností
d) všetky symboly, ktoré sa vyskytujú na pozícií 0 až i niektorého z triedených reťazcov

140. Ktoré z uvedených nádob B[], využívaných algoritmom Radix sort majú uvedený správny obsah, ak j=0 (prvý znak reťazca) a postupnosť pre utriedenie je: 041 145 169 281 334 358 464 467 478 491 500 705 724 827 961 962?

a) B[0]: 500
b) B[8]: 827 //1,5b
c) B[4]: 464 467 478
d) B[3]: 334 385
e) B[7]: 705 724 //1,5b

166. V rámci uvedeného algoritmu triedenia Radix sort, hodnota j predstavuje:

```
begin
   vlož A1, A2, ... , An do QUEUE;
   for j ← k step -1 until 1 do
   begin
      for l ← 0 until m-1 do vyprázdni Q[l];
      while QUEUE ≠ empty do
         begin
            nech Ai je prvý prvok v QUEUE;
            prenes Ai z QUEUE do Q[aij];
         end
      for l ← 0 until m-1 do
         vlož Q[l] na koniec QUEUE;
   end
end
```

a) rozsah hodnôt jednotlivých prvkov (symbolov) reťazca
b) počet triedených reťazcov
c) nič z uvedených možností
d) pozíciu aktuálne spracovaného symbolu v rámci reťazca //3b
e) dĺžku triedených reťazcov

170. Asymptotická zložitosť algoritmu Radix sort pre triedenie reťazcov rôznej dĺžky je daná vzťahom $O(l_{total} + m)$. Hodnota m v tomto vzťahu reprezentuje:

a) počet triedených reťazcov
b) priemernú dĺžku triedených reťazcov
c) maximálne dĺžku triedených reťazcov
d) počet prvkov abecedy triedených reťazcov //3b

194. Asymptotická zložitosť algoritmu Radix sort pre triedenie reťazcov rôznej dĺžky je daná vzťahom $O(l_{total} + m)$. Hodnota l_{total} v tomto vzťahu reprezentuje:

a) počet triedených reťazcov
b) sumu dĺžok triedených reťazcov //3b
c) maximálne dĺžku triedených reťazcov
d) počet prvkov abecedy triedených reťazcov

208. Údajová štruktúra LENGTH[i] (zoznam) využívaná v rámci algoritmu Radix sort pre triedenie reťazcov rôznej dĺžky obsahuje:

a) žiadna z uvedených možností
b) iba reťazce s dĺžkou i //3b
c) iba reťazce s dĺžkou i a menšou
d) iba reťazce s dĺžkou i a väčšou

217. V rámci uvedeného algoritmu triedenia Radix sort, hodnota n predstavuje:

```
begin
   vlož A1, A2, ... , An do QUEUE;
   for j ← k step -1 until 1 do
   begin
      for l ← 0 until m-1 do vyprázdni Q[l];
      while QUEUE ≠ empty do
         begin
            nech Ai je prvý prvok v QUEUE;
            prenes Ai z QUEUE do Q[aij];
         end
      for l ← 0 until m-1 do
         vlož Q[l] na koniec QUEUE;
   end
end
```

a) rozsah hodnôt jednotlivých prvkov (symbolov) reťazca
b) počet triedených reťazcov //3b
c) nič z uvedených možností
d) pozíciu aktuálne spracovaného symbolu v rámci reťazca
e) dĺžku triedených reťazcov

220. Časová zložitosť uvedeného algoritmu Radix sort je najpresnejšie vyjadrená vzťahom:

```
begin
   vlož A1, A2, ... , An do QUEUE;
   for j ← k step -1 until 1 do
   begin
      for l ← 0 until m-1 do vyprázdni Q[l];
      while QUEUE ≠ empty do
         begin
            nech Ai je prvý prvok v QUEUE;
            prenes Ai z QUEUE do Q[aij];
         end
      for l ← 0 until m-1 do
         vlož Q[l] na koniec QUEUE;
   end
end
```

a) T(n) = O(m+n)
b) T(n) = O(kn)
c) T(N)= O((m+n)+k) //3b
d) T(n) = O(km+n)

Bubble:

54. Uvedená postupnosť bola utriedená algoritmom:

2 5 3 8 4 6 7 1 9 (VSTUP)
9 2 5 3 8 4 6 7 1 (l = 1)
9 8 2 5 3 7 4 6 1 (l = 2)
9 8 7 2 5 3 6 4 1 (l = 3)
9 8 7 6 2 5 3 4 1 (l = 4)
9 8 7 6 5 2 4 3 1 (l = 5)
9 8 7 6 5 4 2 3 1 (l = 6)
9 8 7 6 5 4 3 2 1 (l = 7)
9 8 7 6 5 4 3 2 1 (l = 8)
9 8 7 6 5 4 3 2 1 (l = 9)

a) Insert sort
b) Radix sort
c) Bubble sort //3b
d) Heap sort

77. Jednoduché triediace algoritmy (Bubble sort) majú najhoršiu zložitosť:

a) $O(n^3)$ pretože najhorší prípad je pre náhodný vstup a vtedy triedenie trvá dlhšie
b) O(n) pretože je potrebné vykonať n výmen na usporiadanie
c) $O(n^2)$ pretože utriedenie jedného prvku trvá O(n) //3b
d) O(1) pretože vždy sa spotrebuje rovnaký čas na utriedenie

84. Uvedená postupnosť bola utriedená algoritmom:

2 5 3 8 4 6 7 1 9 (VSTUP)
1 2 5 3 8 4 6 7 9 (l = 1)
1 2 3 5 4 8 6 7 9 (l = 2)
1 2 3 4 5 6 8 7 9 (l = 3)
1 2 3 4 5 6 7 8 9 (l = 4)
1 2 3 4 5 6 7 8 9 (l = 5)
1 2 3 4 5 6 7 8 9 (l = 6)
1 2 3 4 5 6 7 8 9 (l = 7)
1 2 3 4 5 6 7 8 9 (l = 8)
1 2 3 4 5 6 7 8 9 (l = 9)

a) Bubble sort //3b
b) Heap sort
c) Radix sort
d) Insert sort

Quick:

129. Doplňte chýbajúci fragment kódu algoritmu rozdelenia postupnosti S na S1 a S2US3 (na obrázku) využíva napr. algoritmus Quick sort:

```
begin
   i ← f;
   j ← l;
   while i ≤ j do
      begin
         while A[j] ≥ a and j ≥ f do j ← j - 1;
         while A[i] < a and i ≤ l do i ← i + 1;
      if i < j then
         begin
            A[i] ↔ A[j];
            [    ?    ];
         end
      end
end
```

a) i <-- i + 1; j <-- j - 1 //3b
b) i <-- i + 1; j <-- j + 1
c) i <-- i - 1; j <-- j - 1
d) i <-- i - 1; j <-- j + 1

Heap:

153. Doplňte chýbajúci riadok kódu algoritmu Heap sort (na obrázku):

```
begin
   BUILDHEAP;
   for i ← n step -1 until 2 do
   begin
      zameň A[1] a A[i];
      [       ?       ];
   end
end
```

a) HEAPIFY(1,n-1)
b) HEAPIFY(1,i-1) //3b
c) HEAPIFY(1,i)
d) HEAPIFY(1,n)

186. Zameň chýbajúci riadok kódu algoritmu triedenia Heap sort (na obrázku):

```
begin
   BUILDHEAP;
   for i ← n step -1 until 2 do
   begin
      [     ?     ];
      HEAPIFY(1,i-1);
   end
end
```

a) zameň A[1] a A[i-1]
b) zameň A[1] a A[i] //3b
c) zameň A[1] a A[n-1]
d) zameň A[1] a A[n]

197. Doplňte chýbajúci riadok kódu algoritmu triedenia Heap sort (na obrázku):

```
begin
   BUILDHEAP;
   for i ← n step -1 until 2 do
   begin
      [     ?     ];
      HEAPIFY(1,i-1);
   end
end
```

a) i <-- i + 1; j <-- j - 1 //3b
b) i <-- i + 1; j <-- j + 1
c) i <-- i - 1; j <-- j - 1
d) i <-- i - 1; j <-- j + 1

236. UŠ typu Zlučovateľná halda (Mergeable heap) podporuje efektívne vykonávanie týchto operácií:

a) DELETE //0,75b
b) MIN //0,75b
c) UNION //0,75b
d) INSERT //0,75b

Ceny násobenia a zložitosti:

12. Primárnym kritériom (časovej) efektívnosti algoritmu pracujúceho s údajmi v súboroch je:

a) počet čítaní/zápisov súboru
b) počet čítaní/zápisov bloku //2b
c) počet čítaní/zápisov bajtu

15. Celkový počet presunov prvkov pri triedení n prvkov metódou priameho zlučovania je rovný:

a) $[\log_n n]$
b) $n.[\log_2 n]$ //3b
c) $[\log_2 n]$

24. Aká je logaritmická cena operandu 'i' stroja RAM?

a) l(i)+l(c(i)) //3b
b) žiadna z uvedených
c) l(i)+l(c(i))+l(c(c(i)))
d) l(c(i))+l(c(c(i)))

35. Pri použití hašovania je odstránenie n prvkov (operácia DELETE, separátne reťazenie), v priemernom prípade vykonané v čase:

a) T(n) = O(n.log n)
b) T(n) = O(n) //3b
c) $T(n) = O(n^2)$
d) T(N) = O(log n)

36. Pri použití hašovania je odstránenie n prvkov (operácie MEMBER, separátne reťazenie), v priemernom prípade vykonané v čase:

a) T(n) = O(log n)
b) T(n) = O(n)
c) $T(n) = O(n^2)$ // 3b
d) T(n) = O(n.log n)

42. Časová zložitosť je definovaná ako počet jednotiek času potrebných na spracovanie vstupu veľkosti n. Ak jednotka času je 1ms, vstup akého najväčšieho rozmeru spracuje algoritmus s časovou zložitosťou $T(n) = n^2$ za 1 sekundu?

a) 11
b) 21
c) 31 // 3b
d) 1000

43. Aká je logaritmická cena inštrukcie ADD i stroja RASP umiestnenej v pamäti od adresy j?

a) l(j)+l(c(0))+l(i)+l(c(i)) //3b
b) l(j)+l(c(i))+l(c(c(i)))
c) l(c(0))+l(i)+l(c(i))+l(c(c(i)))
d) l(c(0))+l(i)+l(c(i))

52. Aká je pamäťová náročnosť reprezentácie grafu (G = (V, E)) pomocou incidenčných zoznamov, ak n = card(V), m = card(E)?

a) S(n) = O(m)
b) S(n) = O(n)
c) S(n) = $O(n^2)$
d) S(n) = O(n+m) //3b

61. Celkový počet blokových operácií pri triedení n prvkov metódou priameho zlučovania je rovný (ak b - počet záznamov v jednom bloku):

a) O((log b)/n)
b) O((n.log n)/b) //3b
c) O((log n)/b)
d) O((b.log n)/n)

62. Aká je logaritmická cena inštrukcie MUL *i stroja RAM?

a) l(c(0))+l(i)+l(c(i))+l(c(c(i))) //3b
b) l(i)+l(c(i))+l(c(c(i)))
c) l(c(0))+l(i)+l(c(i))
a) l(i)+l(c(i))

79. Aká je logaritmická cena operandu '*i' stroja RAM?

a) l(i)+l(c(i))
b) žiadna z uvedených
c) l(i)+l(c(i))+l(c(c(i))) //3b
d) l(c(i))+l(c(c(i)))

80. Aká je logaritmická cena inštrukcie STORE *i stroja RAM?

a) l(c(0))+l(c(i))
b) l(i)+l(c(i))+l(c(c(i)))
c) l(c(0))+l(i)+l(c(i)) //3b
a) l(c(i))+l(i)

85. Časová zložitosť je definovaná ako počet jednotiek času potrebných na spracovanie vstupu veľkosti n. Ak jednotka času je 1ms, vstup akého najväčšieho rozmeru spracuje algoritmus s časovou zložitosťou T(n) = n za 1 hodinu?

a) $6x10^4$
b) 1000
c) $3.6x10^6$ // 3b
d) 100

90. Aká je logaritmická cena inštrukcie LOAD =i stroja RAM?

a) l(c(0))+l(i)+l(c(i))
b) l(i) //3b
c) l(i)+l(c(i))
d) l(c(0))+l(i)

95. $O(n^2)$ najhoršiu zložitosť majú triediace algoritmy:

a) Merge sort
b) Quick sort //1,5b
c) Heap sort
d) Bubble sort //1,5b

100. Technika dynamické programovanie umožňuje výpočet minimálnej ceny násobenia n matíc v čase:

a) $O(n^2)$
b) $O(n^3)$ //3b
c) O(n)

101. Ktoré charakteristiky algoritmov využívame pri ich analýze (zložitosť)?

a) priestorová //1,5b
b) implementačná
c) algoritmická
d) časová //1,5b

106. Procedúra SELECT (verzia využívajúca mediány) nájde k-ty najmenší prvok postupnosti s n prvkami za čas:

a) O(n) //3b
b) O(n.log n)
c) O(log n)
d) $O(n^2)$

107. Časová zložitosť je definovaná ako počet jednotiek času potrebných na spracovanie vstupu veľkosti n. Ak jednotka času je 1ms, vstup akého najväčšieho rozmeru spracuje algoritmus s časovou zložitosťou T(n) = n za 1 minútu?

a) $6x10^4$ //3b
b) 1000
c) $6x10^3$
d) 100

109. Algoritmom pre určenie minimálnej ceny násobenia postupnosti matíc vypočítaná cena vyjadruje potrebný počet operácií:

a) odčítania
b) násobenia //3b
c) sčítania
d) delenia

120. Ktoré cenové kritéria využívame pri určovaní zložitosti algoritmov?

a) uniformné //1,5b
b) kvadratické
c) logaritmické //1,5b
d) lineárne

135. Aká je logaritmická cena inštrukcie WRITE *i stroja RASP umiestnenej v pamäti od adresy j?

a) l(i)+l(c(i))+l(c(c(i)))
b) l(j)+l(i)+l(c(i))
c) žiadna z uvedených //3b
d) l(j)+l(i)+l(c(i))+l(c(c(i)))

137. Procedúra SORT (na obrázku) algoritmu Merge sort pre n>1 uvedený počet porovnaní:

```
procedure SORT(i,j):
begin
    if i=j then return Xi;
        else
            begin
                m ← (i+j-1)/2;
                return MERGE(SORT(i,m),SORT(m+1,j))
            end
end
```

a) 2T(n)+n-1
b) T(n/2)+n-1
c) 2T(n/2)+n-1 //3b
d) T(n/3)+n-1

138. S využitím jednoduchého algoritmu pre UF problém na disjunktných množinách, je možné vykonať n-1 operácií UNION v najhoršom prípade v čase:

a) O(n)
b) $O(n^3)$
c) $O(n^2)$
d) O(n.log n) //3b

142. Časová zložitosť realizácie operácie Member na ADT zoznam (s n prvkami) je:

a) O(n!)
b) O(log n)
c) O(1)
d) Žiadna z uvedených možností //3b

152. Priemernú (očakávanú) zložitosť O(n.log n) majú triediace algoritmy:

a) Bubble sort
b) Quick sort //1,5b
c) Merge sort //1,5b
d) Insertion sort

154. Zložitosť rekurzívneho algoritmu využívajúceho dekompozíciu, podľa vety o povahe a význame dekompozície, ak a>c (kde n - rozmer problému, a - počet podproblémov, n/c - rozmer podproblému) bude:

a) $T(n) = O(n^{\log_c a})$ //3b
b) $T(n) = O(n)$
c) $T(n) = O(n.\log n)$
d) $T(n) = O(1)$

155. Koľko porovnaní vyžaduje realizácia operácie MEMBER pri využití binárneho vyhľadávania?

a) $O(n^2)$
b) $O(n)$
c) $O(\log n)$ //3b
d) $O(n.\log n)$

156. Pri použití hašovania je odstránenie n prvkov (operácia MEMBER, separátne reťazenie), v najhoršom prípade vykonané v čase:

a) $T(n) = O(n.\log n)$
b) $T(n) = O(n)$
c) $T(n) = O(n^2)$ //3b
d) $T(N) = O(\log n)$

165. Aká je logaritmická cena inštrukcie STORE i stroja RAM?

a) l(i)+l(c(i))
b) l(c(0))+l(i) //3b
c) l(i)+l(c(i))+l(c(c(i)))
d) l(c(0))+l(i)+l(c(i))

168. Ktoré z uvedených funkcií majú stupeň rastu $O(n^2)$?

a) n.log n
b) n^3-7n^2
c) $5n^2-7n$ //1,5b
d) $n^2/10 + 10^6n$ //1,5b

169. Aká je logaritmická cena inštrukcie MUL i stroja RAM?

a) l(c(0))+l(i)+l(c(i)) //3b
b) l(i)+l(c(i))+l(c(c(i)))
c) l(c(0))+l(i)
a) l(c(0))+l(c(i))

172. Aká je logaritmická cena inštrukcie LOAD *i stroja RAM?

a) l(c(0))+l(i)+l(c(i))+l(c(c(i)))
b) l(i)+l(c(i))+l(c(c(i))) //3b
c) l(c(0))+l(i)+l(c(i))
a) l(i)+l(c(i))

175. Zložitosť rekurzívneho algoritmu využívajúceho dekompozíciu, podľa vety o povahe a význame dekompozície, ak a<c (kde n - rozmer problému, a - počet podproblémov, n/c - rozmer podproblému) bude:

a) $T(n) = O(n^{\log_c a})$
b) T(n) = O(n) //3b
c) T(n) = O(n.log n)
d) T(n) = O(1)

178. Procedúra SELECT2 (verzia bez využitia mediánov) nájde k-ty najmenší prvok postupnosti s n prvkami za čas O(n):

a) v najhoršom prípade
b) v priemernom prípade //3b

214. Bez ohľadu na cenové kritérium určujeme pri analýze algoritmov tieto typy zložitosti:

a) najlepšia
b) priemerná //1,5b
c) najhoršia //1,5b
d) optimálna

219. Nech x_i je maximálne (z hľadiska požadovaného priestoru) číslo uložené v registri r_i počas vykonávania programu P. Potom logaritmická pamäťová zložitosť RAM programu P je daná:

a) l(x_i) toho r_i, ktoré obsahuje najväčšie číslo počas vykonávania programu
b) l(x_i) toho r_i, ktoré obsahuje najmenšie číslo počas vykonávania programu
c) súčtom l(x_i) nad všetkými pamäťovými registrami, vrátane akumulátora //3b
d) súčtom l(x_i) nad všetkými pamäťovými registrami, okrem akumulátora

221. Časová zložitosť je definovaná ako počet jednotiek času potrebných na spracovanie vstupu veľkosti n. Ak jednotka času je 1ms, vstup akého najväčšieho rozmeru spracuje algoritmus s časovou zložitosťou $T(n) = 2^n$ za 10 sekúnd?

a) 13 //3b
b) 11
c) 9
d) 15

255. Pri použití hašovania je vloženie n prvkov (operácia INSERT, separátne reťazenie), v najhoršom prípade vykonané v čase:

a) T(n) = O(n.log n)
b) T(n) = O(n)
c) T(n) = $O(n^2)$ //3b
d) T(N) = O(log n)

ADT zoznam:

19. Súčasťou definície algebraickej špecifikácie ADT sú:

a) elms
b) opns //1,5b
c) eqns //1,5b
d) axms
e) fncs

32. Ktoré z uvedených definícií sú korektnými definíciami operácií (Opns) algebraickej špecifikácie ADT string?

a) MAKE: alph ->string //3b
b) LADD: string string -> string
c) CAT: alph string -> string
d) EMPTY: ->alph

33. Z nasledujúcich ADT vyberte zložené typy:

a) list //0,75b
b) bool
c) integer
d) nat
e) stack //0,75b
f) array //0,75b
g) real
h) queue //0,75b

41. Medzi zložené ADT patria:

a) množina //0,75b
b) real
c) integer
d) záznam //0,75b
e) zoznam //0,75b
f) pole //0,75b

53. Akú hodnotu nadobudne premenná i po vykonaní uvedenej postupnosti operácií ADT fornt (queue)?

```
Q = CreateQueue( 10 );
Enqueue( 5, Q );
Enqueue( 6, Q );
Enqueue( 7, Q );
Dequeue( Q );
Enqueue( 8, Q );
i = Front( Q );
```

a) 7
b) 8
c) 6 //3b
d) 5

70. Ktoré z uvedených sú korektné definície operácií (Opns) algebraickej špecifikácie ADT stack?

a) EMPTY: ->elm
b) TOP: stack->elm //1,5b
c) TOP: stack->stack
d) EMPTY: ->stack //1,5b

81. Určte typ operácie Advance v rámci implementácie ADT zoznam (list) využívajúcej na cvičeniach:

a) TElement Advance(Position P);
b) Position Advance(Position P); //3b
c) Position Advance(List L);
d) void Advance(TElement);

83. Medzi jednoduché ADT patria:

a) bool //1,5b
b) zoznam
c) integer //1,5b
d) pole
e) záznam

88. Operáciu Cut na ADT zoznam (s n prvkami) je možné vykonať c čase:

a) O(n)
b) O(log n)
c) O(n.log n)
d) lepšom ako ľubovoľná z uvedených možností //3b

99. ADT zásobník (stack) ako variant ADT zoznam (list) - operácie vkladania a odoberania prvkov sú realizované na:

a) rôznych stranách zoznamu
b) rovnakej strane zoznamu //3b

105. Z nasledujúcich ADT vyberte statické typy:

a) array //0,75b
b) tree
c) list
d) set
e) record //0,75b
f) real //0,75b
g) integer //0,75b

110. Medzi fundamentálne operácie na ADT množine patria:

a) TOP
b) DELETE //1,5b
c) MAX
d) MIN // 1,5b

112. Pre reprezentáciu ADT grafu sa používa:

a) incidenčný zoznam //1,5b
b) množina prepojení
c) incidenčná matica //1,5b
d) logická schéma

122. Ktoré z uvedených operácií nie sú operáciami ADT nat?

a) TOP //1,5b
b) SUCC
c) LADD //1,5b
d) MUL

136. ADT zoznam (list) umožňuje v čase O(1) vykonanie operácií:

a) Pop
b) Member
c) Cat //1,5b
d) Insert //1,5b

160. Z nasledujúcich ADT vyberte dynamické typy:

a) array
b) tree //1b
c) list //1b
d) set //1b
e) integer
f) real

179. Aká postupnosť prvkov bude vypísaná po vykonaní uvedenej postupnosti operácií ADT front (queue)?

```
Q = CreateQueue( 10 );
Enqueue( 5, Q );
Enqueue( 6, Q );
Front( Q );
Enqueue( 7, Q );
Dequeue( Q );
FrontAndDequeue( Q );
Enqueue( 8, Q );
PrintQueue( Q );
```

a) 5 6 7
b) 6 7 8
c) 5 6
d) 7 8 //3b

185. Súčasťou definície algebraickej špecifikácie ADT sú:

a) eqns: //1,5b
b) axms:
c) fncs:
d) elms:
e) opns: //1,5b

193. Výhody smerníkovo-reprezentovanej ADT zoznam (list) voči reprezentácií poľom sú:

a) Efektívnejšia realizácia operácie Member

b) Efektívnejšia realizácia operácií Cat a Cut //1,5b

c) Menšie pamäťové nároky na uloženie rovnakých dát

d) Pri operáciách pridávania a odoberania nie je potrebné presúvať dáta //1,5b

203. Uvedený kód predstavuje na ADT zoznam (list) operáciu:

```
Position
xxxx( TElement X, List L )
{
Position P;
P = L;
while( P->Next != NULL && P->Next->Element != X )
P = P->Next;
return P;
}
```

a) First

b) FindPrevious //3b

c) Header

d) Advance

209. Aký režim prístupu k prvkom je využívaný v rámci ADT front (queue)?

a) FIFO //3b

b) LIFO

216. Medzi základné operácie ADT zásobník (implementácie z cvičení) patria:

a) Find

b) Top //0,75b

c) Pop //0,75b

d) Front

e) Insert

f) Push //0,75b

g) IsEmpty //0,75b

218. Ktoré z uvedených definícií predstavujú korektné typy operácií v rámci implementácie ADT front (queue) využívanej na cvičeniach?

a) TElement Front(Queue Q); //1,5b

b) TElement Top(Stack S);

c) void Enqueue(TElement X, Queue Q); //1,5b

d) TElement Dequeue(Queue Q);

224. Ktoré z uvedených definícií predstavujú korektné typy operácií v rámci implementácie ADT zásobník (stack) využívanej na cvičeniach?

a) void Push(TElement X, Stack S); //1,5b
b) TElement Push(TElement X, Stack S);
c) TElement Pop(Stack S);
d) void Pop(Stack S); //1,5b

228. Ktoré z uvedených rovností sú platnými axiómami (eqns) algebrickej špecifikácie ADT stack (a:elm, s:stack)?

a) POP(PUSH(a,s))=s //1,5b
b) POP(PUSH(a,s))=a
c) TOP(PUSH(a,s))=s
d) TOP(PUSH(a,s))=a //1,5b

234. ADT zoznam (list) neumožňuje v čase O(1) vykonanie operácií:

a) Žiadna z uvedených //3b
b) Delete
c) Cut
d) Insert

242. Medzi základné operácie ADT front (queue) patria (implementácia z cvičení):

a) Front //0,75b
b) Insert
c) IsEmpty //0,75b
d) Find
e) Dequeue //0,75b
f) Enqueue //0,75b
g) Top

243. Ktoré z uvedených sú korektné definície operácií (Opns) algebraickej špecifikácie ADT nat?

a) SUCC: nat ->nat //1,5b
b) MUL: nat -> nat
c) MUL: nat nat -> nat //1,5b
d) SUCC: ->nat

248. Určte typ operácie RADD v rámci algebraickej špecifikácie ADT string:

a) string string -> string
b) alph string -> string
c) alph alph -> string
d) string alph -> string //3b

251. ADT zoznam nemôže:

a) vypísať svoj obsah v čase O(1) //3b
b) byť utriedený
c) byť prázdny
d) mať smerníky na predchádzajúci aj nasledujúci prvok zoznamu

253. Akú hodnotu nadobudne premenná i po vykonaní uvedenej postupnosti operácií ADT front (queue)?

```
Q = CreateQueue( 10 );
Enqueue( 5, Q );
Enqueue( 6, Q );
FrontAndDequeue( Q );
Enqueue( 7, Q );
Dequeue( Q );
Enqueue( 8, Q );
i = Front( Q );
```

a) 8
b) 7 //3b
c) 5
d) 6

Stromy:

17. Binárny strom reprezentovaný jednorozmerným poľom A=(16,11,10,8,4,9,6,1,2,5), kde ľavý syn uzla A[i] je A[2i], pravý A[2i+1] haldou (maxheap)?

a) nie //3b
b) áno

18. Pri prehľadávaní grafu do šírky sa používa:

a) kostra grafu
b) zásobník
c) front //3b
d) strom

64. Pre štruktúru halda (maxheap) sú pravdivé tvrdenia:

a) Najvhodnejšia implementácia je pomocou poľa //1,5b
b) Je to binárny strom, ktorého listové vrcholy nemusia mať vždy rovnakú hĺbku //1,5b
c) Obsah haldy je možné napísať v usporiadanom tvare v čase O(n)
d) Je to binárny strom, ktorého všetky listové vrcholy majú vždy rovnakú hĺbku
e) najvhodnejšia implementácia je pomocou smerníkovo-reprezentovaného stromu

97. Pre binárny strom na obrázku platí, že hĺbka vrcholov 4 a 7 je:

a) 2 //3b
b) 3
c) 4
d) 1

96. Je binárny strom reprezentovaný jednorozmerným poľom A=(16,11,9,10,5,6,8,1,2,4), kde ľavý syn uzla A[i] je A[2i], pravý A[2i+1] haldou (maxheap)?

a) nie
b) áno //3b

102. Úplný binárny strom o výške h ma počet všetkých vrcholov:

a) $2^{h+1}-1$ //3b
b) $2^{h}-1$
c) 2^{h}
d) 2^{h+1}

143. Úplný binárny strom o výške h ma počet listových vrcholov:

a) $2^{h+1}-1$
b) $2^{h}-1$
c) 2^{h} //3b
d) 2^{h+1}

183. Pre binárny strom na obrázku platí, že výška vrcholu 6 je:

a) 3
b) 1 //3b
c) 2

190. Základné spôsoby prechádzania grafu sú:

a) do výšky
b) do hĺbky //1,5b
c) do dĺžky
d) do kostry
e) do šírky //1,5b

231. V ktorých typoch stromoch je možné použiť pre vyhľadávanie prvku procedúru search?

a) 2-3 strom
b) AVL strom //1,5b
c) BVS //1,5b
d) ternárny strom

238. Pri prehľadávaní grafu do hĺbky sa používa:

a) kostra grafu
b) strom
c) zásobník //3b
d) front

2.3 Strom:

92. Pre 2-3 strom sú pravdivé tieto tvrdenia:

a) každý nelistový vrchol ma 2 alebo 3 synov //1,5b
b) výšky dvoch podstromov každého vrcholu sa líšia najviac o 1
c) každý nelistový vrchol má najviac 3 synov
d) strom je vždy dokonale vyvážený //1,5b

144. 2-3 strom bol vytvorený postupným vkladaním prvkov 1,3,5,8,6,7,9. Ktoré z uvedených vrcholov budú mať rovnakého rodiča ako vrchol?

a) 9 //1,5b
b) 8 //1,5b
c) 5
d) 6

167. Ktoré z tvrdení sú pravdivé pre 2-3 strom T výšky h?

a) počet listov T nie je nižší ako 3^h
b) počet listov T je rovný aspoň h
c) počet listov T nie je vyšší ako 3^h //3b
d) počet listov T je rovný aspoň 2h

174. Ktoré z tvrdení sú pravdivé pre 2-3 strom T výšky h?

a) počet vrcholov T je najviac 3^h
b) počet vrcholov T je najviac $3^{h+1}-1$
c) počet vrcholov T je najviac $(3^{h+1}-1)/2$ //1,5b
d) počet vrcholov T je aspoň $2^{h+1}-1$ //1,5b
e) počet vrcholov T je aspoň $(2^{h+1}-1)/2$

201. Procedúra IMPLANT:

a) zjednotí dva 2-3 stromy do jedného //3b
b) vyváži 2-3 strom
c) rozdelí 2-3 strom T vzhľadom na list a na dva stromy

202. Majme 2-3 strom T, ktorý vznikol postupným vkladaním hodnôt 1,5,8,3,6,9,7,11 do tohto stromu. Aká bude výška (h) stromu T?

a) h = 2
b) h = 4
c) h = 5
d) h = 3 //3b

206. 2-3 strom bol vytvorený postupným vkladaním prvkov postupnosti 1,3,5,8,6,7,9. Ktoré z uvedených vrcholov budú mať rovnakého rodiča ako vrchol 7?

a) 8 //1,5b
b) 9 //1,5b
c) 5
d) 6

213. Pri označení vnútorných vrcholov 2-3 stromu hodnotami L[v] a M[v] platí:

a) L[v] - maximálny prvok podstromu, ktorého koreňom je najľavejší syn //3b
b) M[v] - maximálny prvok podstromu, ktorého koreňom je najpravejší syn
c) L[v] - minimálny prvok podstromu, ktorého koreňom je najľavejší syn
d) M[v] - minimálny prvok podstromu, ktorého koreňom je najpravejší syn

233. 2-3 stromu sú vhodnou ÚŠ pre efektívne spracovanie týchto inštrukcií:

a) INSERT //0,75b
b) SPLIT //0,75b
c) DELETE //0,75b
d) UNION //0,75b

249. 2-3 strom T bol vytvorený postupným vkladaním prvkov postupnosti 1,3,5,8,6,7,9. Aké bude označenie koreňa stromu T hodnotami L[v] a M[v]?

a) 3:6 //3b
b) 1:8
c) 3:7

256. 2-3 strom T, ktorý vznikol postupným vkladaním hodnôt 1,4,7,3,2,9,8 do tohto stromu. Aké bude označenie koreňa stromu T hodnotami L[v] a M[v]?

a) 3:7 //3b
b) 3:9
c) 2:9
d) 2:7

B strom:

27. Pre B-Strom rádu m sú pravdivé tieto tvrdenia:

a) každá cesta z koreňa k listu má rovnakú dĺžku //1,33b
b) koreň je buď listom, alebo má aspoň [m/2] synov
c) koreň je buď listom, alebo má aspoň dvoch synov //1,33b
d) každý uzol okrem koreňa a listov má počet potomkov medzi [m/2] a m //1,33b

181. Majme B-Strom 5. rádu, obsahujúci jediný kľúč s hodnotou 20. Postupne sú do tohto stromu vkladané kľúče 10, 30, 15, 40, 7, 35. Pri vložení ktorého z týchto kľúčov dôjde k zmene výšky stromu?

a) 30
b) 15
c) 40 //3b
d) 7

232. Majme B-Strom 5. rádu, obsahujúci jediný kľúč s hodnotou 20. Postupne sú do tohto stromu vkladané kľúče 10, 30, 15, 40, 7, 35. Aký kľúč z uvedených bude obsahovať koreň stromu po ich vložení?

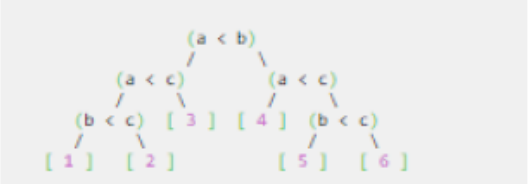
a) 30
b) 20
c) 15 //3b
d) 10

Rozhodovacie stromy:

48. Koľko listov bude mať rozhodovací strom pre utriedenie 3 čísel?

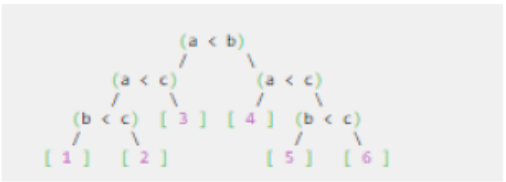
a) 5
b) 8
c) 6 //3b
d) 7

93. Rozhodovací strom pre usporiadanie 3 prvkov a,b,c (na obrázku) obsahuje v liste označenom [5] postupnosť v tvare:



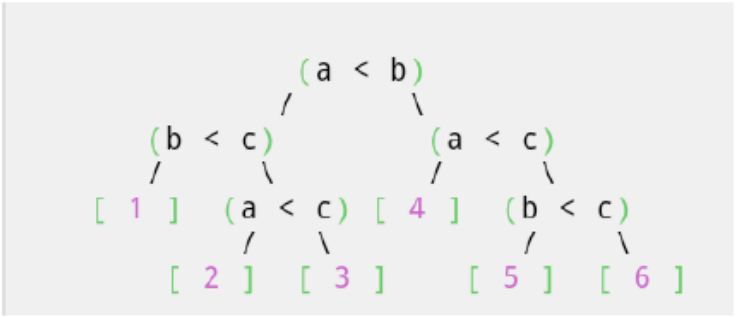
a) a<b<c
b) c<a<b
c) b<a<c
d) b<c<a //3b

145. Rozhodovací strom pre usporiadanie 3 prvkov a,b,c (na obrázku) obsahuje v liste označenom [2] postupnosť v tvare:



a) a<b<c
b) c<a<b
c) b<a<c
d) a<c<b //3b

146. Rozhodovací strom pre usporiadanie 3 prvkov a,b,c (na obrázku) obsahuje v liste označenom [5] postupnosť v tvare:



a) b<a<c
b) a<c<b
c) a<b<c
d) b<c<a //3b

158. Binárny strom reprezentovaný jednorozmerným poľom A=(16,11,10,8,4,9,6,1,5,2), kde ľavý syn uzla A[i] je A[2i], pravý A[2i+1] haldou (maxheap)?

a) nie
b) áno //3b

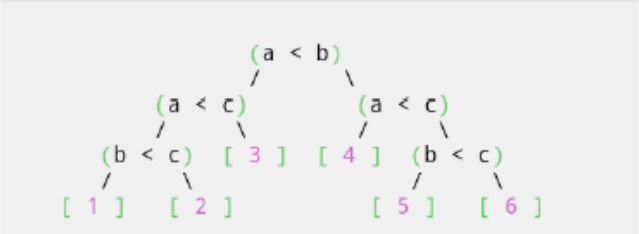
177. Prvky A[i], 1 <= i <= n tvoria haldu (maxheap), ak sú splnené podmienky:

a) A[i] >= A[2i+1], ak 1 <= i <= n/2 //1,5b
b) A[i] >= A[2i+2], ak 1 <= i <= n/2
c) A[i] >= A[2i], ak 1 <= i <= n/2 //1,5b
d) A[i] >= A[i+1], ak 1 <= i <= n/2

180. Akú výšku ma rozhodovací strom pre n prvkov?

a) O(n.log n)
b) O(1)
c) O(n)
d) O(log n) //3b

254. Ktoré z listov rozhodovacieho stromu pre usporiadanie 3 prvkov a,b,c (na obrázku) obsahujú ako posledný (najväčší prvok) postupnosti b?



a) [6]
b) [5]
c) [2] //1,5b
d) [1]
e) [4]
f) [3] //1,5b

BVS strom:

05. Binárny vyhľadávací strom (BVS) je ÚŠ (údajová štruktúra) vhodná pre efektívne vykonávanie operácií:

a) ENQUEUE
b) DELETE //1,5b
c) MIN //1,5b
d) FIND

10. Procedúra ROOTS() pre Výpočet hodnôt $r_{i,j}$ a $c_{i,j}$ Optimálneho BVS využíva techniku:

a) Rekurzia
b) Dynamické programovanie //3b
c) Divide and Conquer
d) Balancing

45. Strom BVS bol vytvorený postupným vkladaním prvkov postupnosti 4, 11, 9, 10, 5, 6, 8, 1, 2, 16. Ktoré z uvedených prvkov postupnosti budú priradené listom vytvoreného stromu?

a) 2 //0,75b
b) 16 //0,75b
c) 10 //0,75b
d) 8 //0,75b

91. Strom BVS bol vytvorený postupným vkladaním prvkov postupnosti 3,5,1,8,6,7,9. Ktoré z uvedených prvkov postupnosti budú priradené listom vytvoreného stromu?

a) 1 //1b
b) 6
c) 5
d) 7 //1b
e) 9 //1b

108. Procedúra BUILDTREE() pre konštrukciu Optimálneho BVS využíva techniku:

a) Rekurzia //3b
b) Greedy
c) Dynamické programovanie
d) Vyvažovanie

162. Údajová štruktúra Optimálny BVS je vhodná pre efektívne spracovanie operácií:

d) MEMBER //3b

131. Strom BVS bol vytvorený postupným vkladaním prvkov postupnosti 3,5,1,8,6,7,9. Ktoré z vrcholov budú mať práve jedného potomka?

a) 5 //1,5b
b) 8
c) 3
d) 6//1,5b

182. Strom BVS bol vytvorený postupným vkladaním prvkov postupnosti 4, 11, 9, 10, 5, 6, 8, 1, 2, 16. Koľko synov bude mať vrchol označený prvkom 5?

a) 1 //3b
b) 0
c) 2
d) 3

188. Koľko listov bude mať rozhodovací strom pre utriedenie 4 čísel?

a) 8
b) 12
c) 24 //3b
d) 16

215. Binárny vyhľadávací strom (BVS) je údajová štruktúra vhodná pre efektívne spracovanie operácií:

a) MEMBER //0,75b
b) INSERT //0,75b
c) MIN //0,75b
d) DELETE //0,75b

AVL strom:

09. AVL strom bol vytvorený postupným vkladaním prvkov postupnosti 1,3,5,8,6,7,9. Ktorý z uvedených prvkov je jeho koreňom?

a) 8
b) 6 //3b
c) 7
d) 3
e) 5

59. AVL strom je vyvážený vtedy a len vtedy, ak:

a) ak výšky dvoch podstromov každého vrcholu sú rovnaké
b) ak výšky dvoch podstromov každého vrcholu sa líšia najviac o 2
c) výšky dvoch podstromov každého vrcholu sa líšia najviac o 1 //3b

82. AVL strom bol vytvorený postupným vkladaním prvkov postupnosti 1,3,5,8,6,7,9. Koľko synov bude mať vrchol označený prvkom 5?

a) 2
b) 0 //3b
c) 1
d) 3

103. AVL strom bol vytvorený postupným vkladaním prvkov postupnosti 1,3,5,8,6,7,9. Ktoré z uvedených prvkov je jeho listami?

a) 3
b) 7 //0,75b
c) 8
d) 6
e) 5 //0,75b
f) 9 //0,75b
g) 1 //0,75b

184. AVL strom bol vytvorený postupným vkladaním prvkov postupnosti 1,3,5,8,6,7,9. Ktoré z vrcholov budú mať práve dvoch potomkov?

a) 7
b) 6 //1b
c) 5
d) 3 //1b
e) 8 //1b

200. Ktorý z uvedených typov rotácií sa využívajú pre vyvažovanie AVL stromov?

a) RR //0,75b
b) LR //0,75b
c) LL //0,75b
d) RL //0,75b

241. Pri realizácií operácie vyvažovania AVL stromu sú posuvy podstromov:

a) iba horizontálne
b) iba vertikálne //3b
c) horizontálne aj vertikálne

Hašovanie:

50. V rámci vzťahu h(k)=[m(kA-[kA])] pre výpočet hodnoty hašovacej funkcie, hodnota A predstavuje:

a) konštantu (z množiny R^+) //3b
b) rozmer hašovacej tabuľky
c) počet prvkov v hašovacej tabuľke
d) kľúč

57. Dobrá hašovacia funkcia by mala mať tieto vlastnosti:

a) nízka miera kolízií //1,5b
b) využívajúca operácia delenia
c) nízka zložitosť výpočtu //1,5b
d) vysoká miera kolízií
e) vysoká zložitosť výpočtu

76. Koľko krát sú urýchlené operácie s kľúčom v prípade hašovaných súborov v porovnaní s jednoduchou organizáciou súboru (zoznam blokov):

a) b-krát, kde b - počet záznamov v bloku
b) n-krát, kde n - počet záznamov
c) žiadne urýchlenie
d) k-krát, kde k - rozmer hašovacej tabuľky //2b

176. Koľko krát sú urýchlené operácie bez kľúča v prípade hašovaných súborov v porovnaní s jednoduchou organizáciou súboru (zoznam blokov):

a) b-krát, kde b - počet záznamov v bloku
b) n-krát, kde n - počet záznamov
c) žiadne urýchlenie //2b
d) k-krát, kde k - rozmer hašovacej tabuľky

78. Hašovanie je technika vhodná pre efektívne vykonávanie operácií:

a) INSERT //1,5b
b) DELETE //1,5b
c) FIND
d) MIN
e) CAT

118. Ktoré zo zoznamov hašovacej tabuľky ostanú prázdne po vložení kľúčov "123", "132", "984", "386", "524", "718" pri použití uvedenej implementácie hašovacej funkcie, ak parameter htsize = 10?

```
unsigned int Hash(char *key,int htsize){
    unsigned int i, value=0;
    for(i=0;i<strlen(key);i++)
        value = (value*10)+(key[i]-'0');
    return value%htsize;
}
```

a) zoznam pre hodnotu 5 //1b
b) zoznam pre hodnotu 4

c) zoznam pre hodnotu 0 //1b
d) zoznam pre hodnotu 1 //1b
e) zoznam pre hodnotu 3
f) zoznam pre hodnotu 2

128. V rámci vzťahu h(k)=k mod m pre výpočet hodnoty hašovacej funkcie, hodnota m predstavuje:

a) konštantu (z množiny R^+)
b) rozmer hašovacej tabuľky //3b
c) počet prvkov v hašovacej tabuľke
d) kľúč

173. Pri použití metódy otvoreného adresovania pre riešenie kolízií hašovania sú jednotlivé kľúče umiestnené:

a) v samotnej hašovacej tabuľke //3b
b) v zoznamoch zodpovedajúcich hodnote hašovacej funkcie
c) v zoznamoch zodpovedajúcich hodnote jednotlivých kľúčov

198. Ktoré z kľúčov "123", "132", "984", "386", "524", "718" vložených do hašovacej tabuľky sa budú nachádzať v rovnakom zozname pri použití uvedenej implementácie hašovacej funkcie, ak parameter htsize = 10?

```
unsigned int Hash(char *key,int htsize){
    unsigned int i, value=0;
    for(i=0;i<strlen(key);i++)
        value = (value*10)+(key[i]-'0');
    return value%htsize;
}
```

a) "386"
b) "123"
c) "524" //1,5b
d) "132"
e) "984" //1,5b
f) "718"

205. Hašované súbory. Aký je priemerný počet blokových prístupov pri vyhľadávaní záznamu podľa kľúča, ak a - počet záznamov, b - počet záznamov v bloku, c -rozmer hašovacej tabuľky?

a) b/ac
b) a/bc //3b
c) c/ab

222. Pojem kolízia v rámci použitia hašovacích funkcií je vyjadrený výrazom:

a) k1 != k2 a h(k1) = h(k2) //3b
b) k1 != k2 a h(k1) != h(k2)
c) k1 = k2 a h(k1) = h(k2)
d) k1 = k2 a h(k1) != h(k2)

230. Pri použití metódy separátneho reťazenia pre riešenie kolízií hašovania sú jednotlivé kľúče umiestnené:

a) v samotnej hašovacej tabuľke
b) v zoznamoch zodpovedajúcich hodnote hašovacej funkcie //3b

Behy:

30. Koľko behov (runs) obsahuje postupnosť **16 01 12 08 07 04 06 11 02 05 10 03 09 12 15 14**?

a) 6
b) 4
c) 8 //2b
d) 7

130. V rámci ktorých postupností z uvedených sú správne vyznačené behy (runs)?

a) 16' 01 12' 08' 07' 04 06 11' 02 05 10' 03 09 13 15' 17
b) 16' 01 12' 08 07' 04 06 11' 02 05 10' 03 09 13 15' 14
c) 16' 01 12' 08' 07' 04 06 11' 02 05 10' 03 09 13 15' 14 //1b
d) 18' 01 16' 08 17' 04 06 11' 02 05 10' 03 09 13 15' 12 //1b

195. Určte počty behov, pre ktoré je možné vykonať korektnú distribúciu počiatočných behov pre optimálnu činnosť algoritmu 3-páskového polyfazového triedenia (bez pridávania tzv. fiktívnych behov)?

a) 13 //15b
b) 21 //1,5b
c) 15
d) 31

229. Nech súbor obsahuje 32 prvkov sformovaných do 10 behov. Koľko prechodov je potrebných na utriedenie prvkov súboru metódou prirodzeného zlučovania?

a) 6
b) 3
c) 5
d) 4 //2b

Matice:

01. Pri násobení dvoch matíc rozmerov [10,20] a [20,1] je rozmer výslednej matice a potrebný počet operácií násobenia:

a) rozmer [20,10] a 2000 operácií
b) rozmer [20,30] a 400 operácií
c) rozmer [10,1] a 200 operácií //3b
d) rozmer [1,10] a 4000 operácií

03. Uvedená matica je incidenčnou maticou grafu:

-	1	2	3	4
1	0	0	1	0
2	1	0	1	0
3	0	0	0	1
4	1	0	0	0

a) orientovaného //3b
b) neorientovaného

46. Jednoduchý algoritmus pre UF problém na disjunktných množinách využíva na reprezentáciu množín UŠ:

a) zásobník
b) front
c) zoznam //3b
d) strom

58. Ktorá z uvedených postupností hrán v grafe zadanom incidenčnou maticou predstavuje kostru grafu vytvorenú jeho prehľadávaním do šírky?

-	0	1	2	3
0	0	1	1	1
1	1	0	0	1
2	1	0	0	1
3	1	1	1	0

a) (0,1),(0,2),(0,3) //3b
b) (0,1),(1,3),(0,2)
c) (0,1),(1,3),(3,2)
d) (0,1),(0,3),(1,3)

75. Minimálne cena násobenia 4 matíc s rozmermi danými hodnotami10, 20, 40, 1, 100 je:

a) 1800
b) 2000 //3b

147. Pri násobení dvoch matíc rozmerov [20,50] a [50,100] je rozmer výslednej matice a potrebný počet operácií násobenia:

a) rozmer [50,50] a 10000 operácií
b) rozmer [50,125] a 1000 operácií
c) rozmer [20,100] a 100000 operácií //3b
d) rozmer [50,100] a 10000 operácií

161. Objekt akej povahy je výstupom činnosti algoritmu pre určenie minimálnej ceny násobenia matíc?

a) jedna matica
b) neprázdna postupnosť čísel
c) neprázdna postupnosť matíc
d) jedno číslo //3b

199. Ktoré z uvedených dvojíc predstavujú rozmery matíc spracovaných algoritmom pre určenie minimálnej ceny násobenia matíc, ak vstup algoritmu je daný poľom rozmer[]?

```
int rozmery[]={ 10, 40, 30, 60, 20, 50 };
```

a) (30,20)
b) (40,30) //1,5b
c) (30,40)
d) (20,50) //1,5b

210. Koľko matíc je reprezentovaných uvedeným vstupom algoritmu pre určenie minimálnej ceny násobenia matíc?

```
int rozmery[]={ 10, 40, 30, 60, 20, 50 };
```

a) 5 //3b
b) 3
c) 6
d) 7

258. Minimálne cena násobenia 4 matíc s rozmermi danými hodnotami 10, 20, 30, 1, 100 je:

a) 1800 //3b
b) 2000
c) 2100
d) 1900

Iné:

06. Vyvážené viaccestné zlučovanie pri použití N pások realizuje:

a) N - cestné zlučovanie
b) N-2 - cestné zlučovanie
c) N/2 - cestné zlučovanie //2b
d) N-1 - cestné zlučovanie

07. Medzi relácie čiastočného usporiadanie (partial order) patria:

a) <= na Z //1,5b
b) $\subseteq$ na množinách //1,5b
c) < na Z
d) > na Z

23. Problém zhlukovania (Clustering) pri riešení kolízii stratégiou otvorenej adresácie sa v najvyššej miere vyskytuje pri použití:

a) kvadratického testovania (quadratic probing)
b) dvojitého hašovania (double hashing)
c) lineárneho testovania (linear probing) //3b

28. Zásobníkový rámec pri volaní procedúr obsahuje:

a) Adresa začiatku volajúcej procedúry
b) Priestor pre lokálne premenné //1,5b
c) Aktuálne parametre //1,5b
d) Meno volajúcej procedúry

47. Vonkajšia pamäť je rozdelená z pohľadu OS na bloky rovnakej veľkosti. Typická veľkosť takých blokov je:

a) 512MB - 4GB
b) 512kB - 4MB
c) 512B - 4kB //3b

121. Vyvážené viaccestné zlučovanie pri použití N pások realizuje:

a) N-2 - cestné zlučovanie
b) N-1 - cestné zlučovanie
c) N/2 - cestné zlučovanie //2b
d) N - cestné zlučovanie

125. Riedky (sparse) index pozostáva z párov:

a) (x,b), kde b - adresa bloku, v ktorom prvý záznam má kľúč x //3b
b) (x,p), kde p - smerník na záznam s kľúčom x
c) (x,f), kde f - súbor, v ktorom sa nachádza záznam s kľúčom x

139. Pri použití relácie lexigrafického usporiadania platí:

a) 2223 <= 223 //1,5b
b) 59 <= 123
c) 123 <= 59 //1,5b
d) 223 <= 2223

55. Ktoré z uvedených techník organizovania súborov umožňujú sprístupnenie prvkov v utriedenom poradí?

a) riedky index //1b
b) hašovanie
c) B-strom //1b

141. Riedky index možno využiť, ak:

a) súbor je utriedený podľa hodnoty kľúča //2b
b) súbor nie je utriedený podľa hodnoty kľúča

250. Pri súboroch bez usporiadania záznamov podľa kľúča, je možné využiť pre urýchlenie vyhľadávania:

a) riedky index
b) hustý index //2b

149. Pre polyfázové triedenie pri použití N pások je charakteristické:

a) N - cestné zlučovanie
b) N-2 - cestné zlučovanie
c) N/2 - cestné zlučovanie
d) N-1 - cestné zlučovanie //3b

163. Smerník na záznam v súbore (tzv. pripichnuté záznamy) obsahuje:

a) offset (počet bajtov v bloku pred začiatkom záznamu) //1b
b) fyzickú adresu (začiatku) bloku na sekundárnom médiu //1b
c) veľkosť záznamu (v bajtoch)
d) počet nasledujúcich záznamov rovnakého typu

204. Relácia čiastočného usporiadania (partial order) na množine S je:

a) tranzitívna //1b
b) ireflexívna
c) reflexívna //1b
d) antisymetrická //1b
e) symetrická

211. Štruktúry zložené z prvkov rovnakého typu sú:

a) zoznam //1b
b) pole //1b
c) štruktúra
d) záznam
e) strom //1b

223. Hustý (dense) index pozostáva z párov:

a) (x,b), kde b - adresa bloku, v ktorom prvý záznam má kľúč x
b) (x,p), kde p - smerník na záznam s kľúčom x //3b
c) (x,f), kde f - súbor, v ktorom sa nachádza záznam s kľúčom x

237. Procedúra SELECT(verzia využívajúca mediány) realizuje delenie postupnosti S na 3 časti (S1, S2, S3) vzhľadom na medián m. Maximálny rozmer podpostupností S1 (respektíve S3) je:

a) (1/4)n
b) (1/2)n
c) (2/3)n
d) (3/4)n //3b

240. Procedúra SELECT(verzia využívajúca mediány) realizuje delenie postupnosti S na 3 časti (S1, S2, S3) vzhľadom na medián m. Počet prvkov S <= m je aspoň:

a) (3/4)n
b) (1/2)n
c) (2/3)n
d) (1/4)n //3b

246. Pri použití relácie lineárneho usporiadania platí:

a) 2223 <= 223
b) 59 <= 123 //1,5b
c) 123 <= 59
d) 223 <= 2223 //1,5b