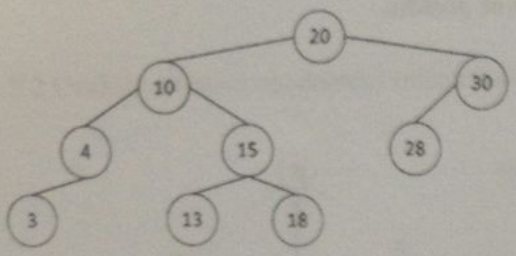
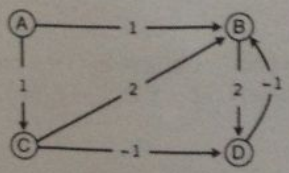


L 2 Toto je pole práve po prvom rozčlenení v algoritme rýchleho usporadúvania (quicksort):
4, 1, 3, 5, 6, 9, 8, 7, 10
Ktorý z týchto prvkov mohol byť pivot? (môže ich byť viac takých!)

M 3 Bellman-Ford Uvažujte orientovaný graf s ohodnotenými hranami, ktorý má aj záporne ohodnotené hrany, avšak nemá záporné cykly. Dokončite stopu výpočtu Bellmanovho-Fordovho algoritmu, ktorý sa začne vykonávať vo vrchole A, ako je znázornená v tabuľke nižšie. Ako pomôcka je uvedená stopa prvej fázy relaxácie. (successful relax = úspešná relaxácia hrany, shortest known distance to = najkratšia doteraz známa vzdialenosť do). Hrany sa v každej fáze relaxujú v poradí, v akom sú uvedené pod sebou v tabuľke.



		successful relax?	shortest known distance to			
			A	B	C	D
			0			
phase 1	D→B					
	C→D					
	C→B					
	B→D					
	A→C	X			1	
	A→B	X		1		
phase 2	D→B					
	C→D	X				0
	C→B			3		
	B→D					3
	A→C				1	
	A→B			1		
phase 3	D→B	X		-1		
	C→D					0
	C→B			3		
	B→D					31
	A→C				1	
	A→B			1		
phase 4	D→B			-1		
	C→D					0
	C→B			3		
	B→D					+1
	A→C				1	
	A→B			1		

Dátové štruktúry a algoritmy piatok, 10. januára 2014

Podpíšte tento list aj dvojhárok – meno, priezvisko a osobné číslo.
Odpovede píšte priamo na tento list. Pracujete samostatne a odovzdáte len výsledky vlastnej práce, dosiahnuté bez pomoci.
Meno a priezvisko: *osobné číslo:*

A	B	C	D	E	F	G	H	I	J	K	L	M	
2	7	6	4	2	2	3	6	3	3	2	2	3	
2	3	0	4	2	0	2	1	3	3	2	2	3	

A 2 Uvažujte túto verziu algoritmu usporadúvania vkladaním.

INSERTION-SORT(A)

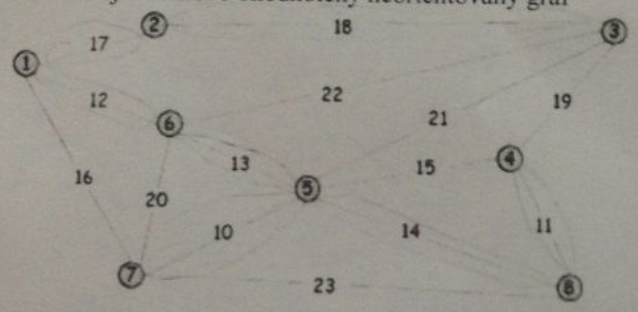
```
1  for j ← 2 to length[A]
2      do key ← A[j]
3         i ← j − 1
4         while i > 0 and A[i] ≥ key
5             do A[i + 1] ← A[i]
6                i ← i − 1
7         A[i + 1] ← key
```

Je tento algoritmus stabilný? Zvoľte jednu z dvoch možností:
Áno – a vysvetlite, prečo je stabilný
alebo
Nie – a vysvetlite, čo ho robí nestabilný a zmeňte ho, aby sa stal stabilný.

B 7 Uvažujte hranovo ohodnotený orientovaný graf G s kladnými ohodnoteniami hrán *w(u,v)* medzi vrcholmi *u* a v. Vrchol s je východiskový. Zmeňte Dijkstrov algoritmus tak, aby počítal aj počet rôznych ciest minimálnej dĺžky z vrchola *s* do každého vrchola *v*.

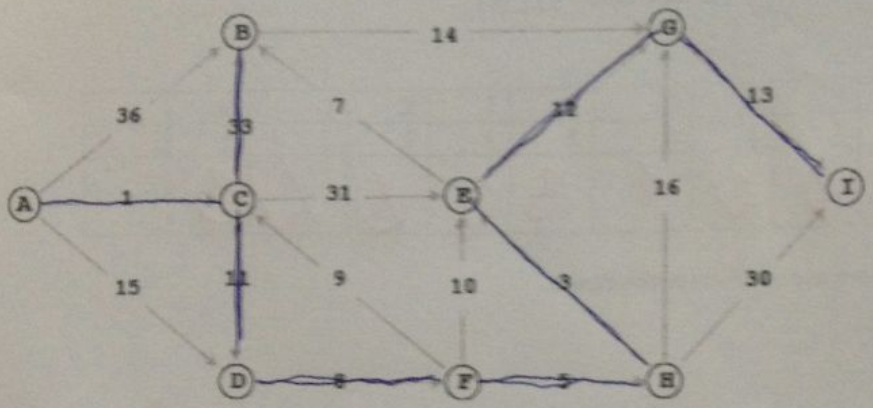
C 6 Daná je postupnosť n čísel a jedno číslo, ktoré nazveme s. Navrhnite algoritmus, ktorý rozhodne v lineárnom čase, či súčet niektorých dvoch čísel v postupnosti je s. Nemusíte písať presne v nejakom programovacom jazyku, stačí pseudokód alebo slovný opis.

D 4 Uvažujte hranovo ohodnotený neorientovaný graf



a) napíšte hrany v minimálnej kostre v poradí, ako ich nájde Primov algoritmus, ktorý začne vo vrchole 1.
b) napíšte hrany v minimálnej kostre v poradí, ako ich nájde Kruskalov algoritmus.

E 2 Uvažujte hranovo ohodnotený orientovaný graf na obrázku.



Ukážte vykonanie Dijkstrovho algoritmu na ňom. Východiskom je vrchol A.

a) Uveďte vrcholy v poradí,v akom sa budú vyberať z min-prioritného frontu. Ku každému vrcholu uveďte dĺžku najkratšej cesty doňho z vrchola A.

vrchol	A	C	D	F	H	E	B	G	I
dĺžka	0	1	12	20	25	28	34	40	53

b) Vyznačte v obrázku tučne hrany, ktoré tvoria najkratšie cesty.

F 2 Uvažujte túto implementáciu usporadúvania zlučovaním:

```
public class Merge
{
   public static void sort(Comparable[] a, Comparable[] aux, int lo, int hi)
   {
      if (hi <= lo) return;
      int mid = lo + (hi - lo) / 2;
      sort(a, aux, lo, mid);
      sort(a, aux, mid+1, hi);
      merge(a, aux, lo, mid, hi); // merges 2 sorted subarrays into a[lo..hi].

      System.out.print(lo + " " + hi + " ");
      for (int i = lo; i <= hi; i++)
         System.out.print(a[i] + " ");
      System.out.println();
   }

   public static void sort(Comparable[] a)
   {
      int N = a.length;
      Comparable[] aux = new Comparable[N];
      sort(a, aux, 0, N-1);
   }
}
```

Všimnite si, že sort je doplnený o vypisovanie hodnôt medzi (indexov) časti poľa, ktoré sa v tom volaní spracúvajú a samotných znakov v ňom. Došlo k jeho zavolaniu:

Character[ ] a = { 'z', 'y', 'x', 'w', 'v', 'u', 't' , 's' , 'r' };
Merge.sort(a);

Výstupné riadky sú nižšie, avšak v prehádzanom poradí. Napíšte správne poradie.

A. 02xyz
B. 34vw
C. 04vwxyz
D. 58rstu
E. 08rstuvwxyz
F. 01yz
G. 78rs
H. 56tu

Tu uveďte písmená, označujúce riadky, v správnom poradí (posledný riadok E sme predvyplnili).

G	H	B	F	A	D	C	E

G 3 Uvažujte binárnu max-haldu v jej reprezentácii v poli:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-	X	W	J	V	U	D	H	S	P	Q	R	C	-

a) Nakreslite zodpovedajúcu haldu v tvare binárneho stromu.

b) Vložte do haldy prvok M. Napíšte, ako bude halda reprezentovaná po vložení a zakrúžkujte hodnoty, ktoré sa zmenili:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-	X	W	M	V	U	J	H	S	P	Q	R	C	D

c) Zmažte maximálny prvok z pôvodnej haldy. Napíšte, ako bude halda reprezentovaná po zmazaní a zakrúžkujte hodnoty, ktoré sa zmenili:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-	W	U	J	V	R	D	H	S	P	Q	C		

H 6 Hľadanie 2D vzoru. Štandardné algoritmy na vyhľadávanie reťazcov (KMP, RK, BM), fungujú pre 1D vzory (lineárny reťazec znakov). 2D vzor nech je matica P1 riadkov a P2 stĺpcov znakov. Vzor vyhľadávame v 2D mape (matici znakov), ktorá má M1 riadkov a M2 stĺpcov. Navrhnite algoritmus – nejakú kombináciu spomínaných 1D algoritmov – ktorý v lineárnom čase O(P1*P1 + M1*M2) alebo rýchlejšie určí, kde sa 2D vzor nachádza v 2D mape. Napíšte hlavnú myšlienku, algoritmus neprogramujte.

I 3 Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je 6 (miest pamäti) a rozptylová funkcia je h(x) = x mod 6, pričom kolízie sa riešia zreťazením. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 35, 2, 18, 6, 3, 10, 8, 5.
(Nekreslite priebežné stavy.)

J 3 Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je 7 (miest pamäti) a rozptylová funkcia je h(x) = x mod 7, pričom kolízie sa riešia lineárnym skúšaním. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 16, 18, 15, 2, 9, 14, 7.

0	1	2	3	4	5	6
14	15	16	2	18	9	7

K 2 Uvažujte AVL strom na obrázku. Nakreslite strom po vložení prvku 17.

PROSIM NEMAZTE NIC! DAKUJEM.

Alebo ked sa neviete zdrzat aby ste daco nezmazali tak napravo hore je prepinac kde pise "Editing" a prepnite ho na "Viewing"! Dakujem

riešenie veľkosť písma 11, postupy ako na to veľkosť písma 8, komentáre veľkosť písma 11, ale iná farba, napr táto.

A

nie

podľa http://www.ee.ryerson.ca/~courses/coe428/sorting/insertionsort.html zmena v štvrtom riadku programu zo skúšky A[i] > key, nestabilným ho robilo to, že dva prvky s rovnakým kľúčom si zmenili poradie v akom boli uložené v poli a v stabilnom algoritme si dva prvky s rovnakým kľúčom poradie zachovávajú.

B

spravis si druhe pole alebo pridas jeden integer do tych vrcholov a vzdy ked prejdes tym vrcholom a je to najkratsia (sucasne!) cesta tak das integer++

tiež pozri: http://www.cs.duke.edu/courses/fall09/cps230/hws/midterm/head.pdf

//neviem co tym sucasne myslis, ale ja by som dal ++ ked sa ta cesta presne rovna tej s ktorou ju prave porovnavam…//+1 no ak je mensia tak to treba initovat zase na 1 (pocitas odznova)

```
//cize nieco taketo, ne (upraveny z wikipedie)?:
1  function Dijkstra(Graph, source):
2      dist[source] ← 0                          // Initialization
3
4      create vertex set Q
5
6      for each vertex v in Graph:
7      if v ≠ source
8              dist[v] ← INFINITY                // Unknown distance from source to v
9              prev[v] ← UNDEFINED               // Predecessor of v
10     cnt[v] ← 1                                // kazdy uzol ma aspon 1 najkratsiu cestu //a co ak je niektory unreachable?
11
12     Q.add_with_priority(v, dist[v])
13
14
15     while Q is not empty:                     // The main loop
16     u ← Q.extract_min()                       // Remove and return best vertex
17     for each neighbor v of u:                 // only v that is still in Q
18             alt = dist[u] + length(u, v)
19             if alt == dist[v]                 // ak rovnako dlha cesta k nemu uz viedla
20               cnt[v]++                        // tak pripocitame
21             if alt < dist[v]
22               dist[v] ← alt
```

```
23                  prev[v] ← u
24                  cnt[v] ← 1                          // ak sa nasla kratsia, tak "vynulovat"
25                Q.decrease_priority(v, alt)
26
27        return dist[], prev[]
```

C

Hashovacia tabuľka
Postupne prechádzať čísla v poli a pozrieť sa do tabuľky (na začiatku je prázdna), či sa tam nenachádza s-n[i], ak nie, tak uložiť číslo do tabuľky a pokračovať ďalším číslom. Ak áno, tak sa číslo našlo, keď sme prešli celé pole, tak sa číslo nenašlo.

D

Primov: 12, 13, 10, 14, 11, 17, 18
Kruskalov: 10, 11, 12, 13, 14, 17, 18

E

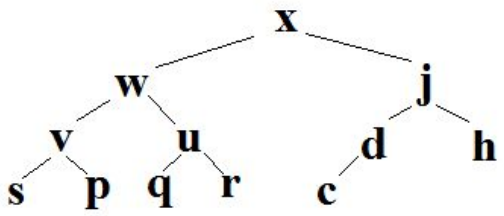
vyriešené
https://www.youtube.com/watch?v=gdmfOwyQlcI

F

F, A, B, C, H, G, D, E
,3

G

a) - X koreň, W, J jeho deti, V U su deti rodica W, potom D H su deti rodica U a takto postupuješ po úrovniach.
//D a H su deti J nie ?
// Nie sú náhodou LCHILD (2 * i + 1) a RCHILD (2 * i + 2)? Potom to nesedí, keďže X je na indexe 1, tak LCHILD by bol na indexe 2 * 1 + 1, teda na treťom… Ako teda??
//Áno bolo to zle, ma to byt takto:



b) - vyriešené
c) -, W, V, J, S, U, D, H, C, P, Q, R, -, - (odstrániš najväčší prvok a namiesto neho dáš posledný a potom vždy porovnávaš s väčším z detí a ak je dieťa väčšie tak ich vymeníš)

H

Rabin-Karp pouzit na premenu mapy a vzoru na jednorozmerne pole cisel, nasledne to pomocou KMP porovnavat
http://theory.snu.ac.kr/mediawiki/images/9/97/Lec6.pdf

I

(podľa prednášky sa nový prvok pri kolízii pridáva na začiatok, ak sa už v danom bucket-e nenachádza)

//myslím, že to závisí od implementácie, či sa pridáva na začiatok alebo nie //+2

0	*1*	*2*	*3*	*4*	*5*
6		8	3	10	5
18		2			35

J, K

vyriešené

L

5, 6, 10

o každom z týchto čísel platí, že čísla pred ním sú menšie a čísla za ním sú väčšie // neplati to aj o 3? //vsetky cisla pred nim musia byt mensie zrejme a pred 3 je aj 4

M

phase 2: C->D x -, -, -, 0

phase 3: D->B x -, -1, -, -

staci mat spravne riadky kde su succesful relax (x-ka), ostatne ich velmi nezaujima.

Dátové štruktúry a algoritmy

piatok, 9. januára 2015

Podpíšte tento list aj dvojhárok – meno, priezvisko a osobné číslo.
Odpovede píšte priamo na tento list. Pracujete samostatne a odovzdáte len výsledky vlastnej práce, dosiahnuté bez pomoci.
Meno a priezvisko: JAROSLAV Z160 *osobné číslo:*

A	B	C	D	E	F	G	H	I	J	K	L	M	N	spolu
2	4	3	5	3	2	1	6	6	4	3	2	2	2	

A 2 Uvažujte algoritmus usporadúvania výberom:

```
procedure SelectionSort(var A: pole)
var i, j, t : integer;
begin
for i:= 1 to Dlzka(A) – 1 do
        MinIndex := i;
        for j:= i to Dlzka(A)- 1 do
                if A[MinIndex] > A[j] then
                        MinIndex := j;
        T := A[i];
        A[i] := A[MinIndex];
        A[MinIndex] := T;
        end;
```

Je tento algoritmus stabilný?

a) Áno, lebo: ZACHOVÁVA PREVY S ROVNAKOU HODNOTOU V ORIGINÁLNOM PORADÍ (PÔVODNOM)
b) Nie, lebo:

Vo zdôvodnení odpovede uveďte, ktorý príkaz/y zaručia/môžu porušiť stabilitu. V prípade odpovede Nie uveďte aj príklad postupnosti, pri usporadúvaní ktorej dôjde k nestabilite.

B 4 Uvažujte rýchle usporadúvanie:

```
QUICKSORT(A, p, r)
1  if p < r
2     then q ← PARTITION(A, p, r)
3          QUICKSORT(A, p, q − 1)
4          QUICKSORT(A, q + 1, r)
```

Napíšte jeho znáhodnenú verziu. Napíšte aj algoritmus rozčleňovania použitého v znáhodnenom rýchlom usporadúvaní.

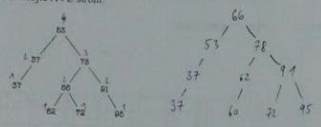
C 3 Uvažujte algoritmus usporadúvania spočítavaním. Používa tri polia: A – v ňom je pôvodná postupnosť n prvkov, B - v ňom bude usporadaná postupnosť a C – tzv. počítadlá.

a) Doplňte dĺžky jednotlivých polí: A[n], B[ ], C[ ].
b) Uvažujte vstupnú postupnosť 2, 3, 1, 2, 3. Napíšte, aký bude obsah všetkých troch polí na konci.

A: B: C:

D 5 Je dané usporiadané pole celých čísel A[1..n]. Navrhnite algoritmus, ktorý rozhodne, či existuje index i taký, že i=A[i]. Ak existuje, najdiindex(A,1,n) vráti i, ak nie, vráti 0. Vaše riešenie musí byť rýchlejšie ako O(n) a nesmie vyžadovať viac dodatočnej pamäti ako O(1).

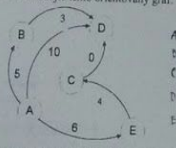
E 3 Uvažujte AVL strom:



Pripíšte ku každému uzlu hodnotu jeho faktoru vyváženia (výška ľavého postromu – výška pravého podstromu)

Vložte do tohto stromu 60 a ak treba, urobte úpravy, aby bol výsledný strom opäť AVL. Nakreslite aj strom po každej úprave.

F 2 Uvažujte tento orientovaný graf:



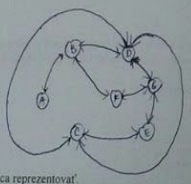
	A	B	C	D	E
A	0	1	0	1	1
B	0	0	0	1	0
C	0	0	0	1	0
D	0	0	0	0	0
E	0	0	1	0	0

A→B, A→D, A→E
B→D
C→D
E→C

Nakreslite jeho reprezentáciu pomocou matice susednosti a pomocou zoznamov susediacich uzlov.

G 1 Uvažujte maticu susednosti:

	A	B	C	D	E	F	G
A	0	1	0	0	0	0	0
B	1	0	0	1	0	1	0
C	0	0	0	1	1	0	0
D	0	1	1	0	0	0	1
E	0	0	1	0	0	0	1
F	0	1	0	0	0	0	1
G	0	0	0	1	1	1	0



Nakreslite graf, ktorý môže táto matica reprezentovať.

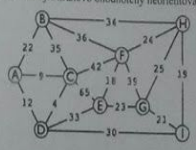
H 6 Ukážte priebeh radixového usporadúvania (radix je 10) tejto postupnosti čísel:
346, 22, 31, 212, 157, 102, 568, 435, 8, 14, 5

I 6 Daná je postupnosť n čísel a jedno číslo, ktoré nazveme s. Navrhnite algoritmus, ktorý rozhodne v lineárnom čase O(n), či súčet niektorých dvoch čísel v postupnosti je s. Nemusíte písať presne v nejakom programovacom jazyku, stačí pseudokód alebo slovný opis.

```
FOR i IN postupnost[]
    pre rozdiel = S - postupnost[i]
    boolean jeRozdielVHashovacej = jeRozdielVHashovacej(rozdiel)
    if (jeRozdielVHashovacej)
        return true
    else
        vlozCisloZPostupnostiDoHashovacej(postupnost[i])
END FOR
```

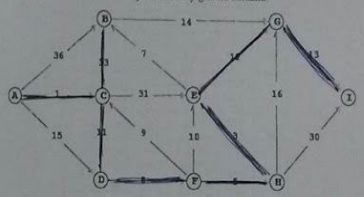
Pozn. Postupne by som z danej postupnosti (z polí) vyťahoval prvky a vyhľadával rozdiel do najväčšieho S. Následne by som hľadal rozdiel v hashovačke. Ak by som ho tam našiel, funkcia skončila, ak nie, pridal by som vyhľadané číslo z postupnosti do hashovačky a pokračoval do ďalšieho cyklu (všetkých prvkov postupnosti).

J 4 Uvažujte hranovo ohodnotený neorientovaný graf



a) napíšte hrany v minimálnej kostre v poradí, ako ich nájde Kruskalov algoritmus. 4, 9, 11, 19, 21, 22, 23, 30
b) napíšte hrany v minimálnej kostre v poradí, ako ich nájde Primov algoritmus, ktorý začne vo vrchole A. 9, 4, 22, 30, 21, 23, 11, 25

K 3 Uvažujte hranovo ohodnotený orientovaný graf na obrázku.

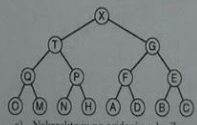


Ukážte vykonanie Dijkstrovho algoritmu na ňom. Východiskom je vrchol A.
Uveďte vrcholy v poradí, v akom sa budú vyberať z min-prioritného frontu. Ku každému vrcholu uveďte dĺžku najkratšej cesty doňho z vrchola A.

vrchol	A	C	D	F	H	E	B	G	I
vzdialenosť	0	1	12	20	25	28	33	40	53

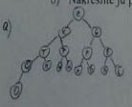
Vyznačte v obrázku tučne hrany, ktoré tvoria najkratšie cesty.

L 2 Uvažujte max-haldu :

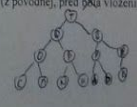


a) Nakreslite ju po pridaní znaku Z.
b) Nakreslite ju po odobratí maxima (z pôvodnej, pred tým vložením Z).

a)



b)

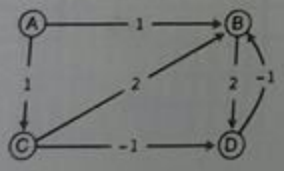


M 2 Predpokladajme, že sa haldou usporadúva pole siedmich celých čísiel. Stav po nejakom vyhaldovaní (heapify) je:
7 5 6 2 3 8 9
Koľko vyhaldovaní už prebehlo? (KOĽKO NAJVIAC?) ☺

A. 1
B. (2)
C. 3 alebo 4
D. 5 alebo 6

N 2 Bellman-Ford Uvažujte orientovaný graf s ohodnotenými hranami, ktorý má aj záporne ohodnotené hrany, avšak nemá záporné cykly. Dokončite stopu výpočtu Bellmanovho-Fordovho algoritmu, ktorý sa začne vykonávať vo vrchole A, ako je znázornená v tabuľke nižšie. Ako pomôcka je uvedená stopa prvej fázy relaxácie. (successful relax = úspešná relaxácia hrany, shortest known distance to = najkratšia doteraz známa vzdialenosť do). Hrany sa v každej fáze relaxujú v poradí, v akom sú uvedené pod sebou v tabuľke.

		successful relax?	shortest known distance to			
			A	B	C	D
			0			
phase 1	D→B					
	C→D					
	C→B					
	B→D					
	A→C	X			1	
	A→B	X		1		
phase 2	D→B					
	C→D					
	C→B					
	B→D	X				2
	A→C	[illegible]		[illegible]	[illegible]	
	A→B	[illegible]		[illegible]		
phase 3	D→B					
	C→D	X				-1
	C→B	X		2		
	B→D					
	A→C					
	A→B					
phase 4	D→B	X		-1		
	C→D					
	C→B					
	B→D					
	A→C					
	A→B					



f

A

http://stackoverflow.com/questions/20761396/why-selection-sort-can-be-stable-or-unstable

b) Nie, toto je implementacia nestabilneho , nestabilnym ho robi swapovanie v predposlednom riadku.
Priklad postupnosti(1 4 4 2): [i=0] 1 4(1.) 4(2.) 2 -> [i=1] 1 4(1.) 4(2.) 2 -> [i=2] 1 2 4(2.) 4(1.) ...
Poznámka: Zaujimavé, že v prednáške je uvedené, že selection je stabilný, pričom to kód v skúške vyvracia. Záleží to od implementácie.

B

http://www.albany.edu/~csi503/pdfs/handout_8.1.pdf

Znáhodnené rýchle usporadúvanie

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2     then q ← RANDOMIZED-PARTITION(A, p, r)
3          RANDOMIZED-QUICKSORT(A, p, q − 1)
4          RANDOMIZED-QUICKSORT(A, q + 1, r)
```

```
RANDOMIZED-PARTITION(A, p, r)
1  i ← RANDOM(p, r)
2  exchange A[r] ↔ A[i]
3  return PARTITION(A, p, r)
```

C

http://www.albany.edu/~csi503/pdfs/handout_9.1.pdf

a) A[n], B[n], C[max(A[ ])+1] //max(n)+1 je blbosť, má to byť maximálny prvok z poľa A, pseudozápis napr max(A[ ]) //nema to byt pocet rozlicnych klucov, v pripade ak by bol kazdy kluc iny tak n

Usporadúvanie spočítavaním

- Pracuje s tromi poliami:
  - Pole *A[]* obsahuje údaje, ktoré sa majú usporiadať – veľkosť poľa je *n*
  - Pole *B[]* obsahuje konečný usporiadaný zoznam údajov – veľkosť poľa je *n*
  - Pole *C[]* je použité na počítanie počtu prvkov – veľkosť poľa je *k*

vstup bol: 2,3,1,2,3

b) A:2 3 1 2 3 B:1 2 2 3 3 C: 0 1 3

https://www.youtube.com/watch?v=5rLrRpcBCzo

// pozn: pomocne pole indexujte od 1 nie 0

// Nema to byt 0 1 3 5? -+1

D

poznámka: Prvky sa neopakujú, taktiež sa nemôže použiť Binárne vyhľadávanie rekurziou nakoľko by sa zvýšila pamäťová náročnosť.

Binárne vyhľadávanie s cyklom

```
function Search(List: IntList; Key: Integer)
return Integer
        Low: Integer := List'First;
        High: Integer := List'Last;
        Mid: Integer;
        begin
        loop
                Mid := (Low + High) / 2;
                if Key = A[Mid] then return Mid;
                        elsif Key < A[Mid] then High := Mid - 1;
                        else Low := Mid + 1;
                end if;
                if Low > High then return 0;
        end loop;
end Search;
```

Úprava tejto implementácie, v parametroch funkcie nepotrebuješ Key a v tele funkcie nahradis Key Mid-om. Zo zadania A znamená IntList a ListFirst by bol druhý parameter, ListLast by bol tretí parameter.

príklad Java:

```java
    // int jednotka = 1

    public int najdiindex(int A[], int jednotka, int n) {
        int dolnyIndex = jednotka;
        int hornyIndex = n;
        int i;
        while (true) {
            i = (dolnyIndex + hornyIndex) / 2;
            if (i == A[i])
                return i;
            else if (i < A[i]){
                hornyIndex = i - 1;
            }else{
                dolnyIndex = i + 1 ;
            }
            if (dolnyIndex > hornyIndex)
                return 0;
        }
    }
```

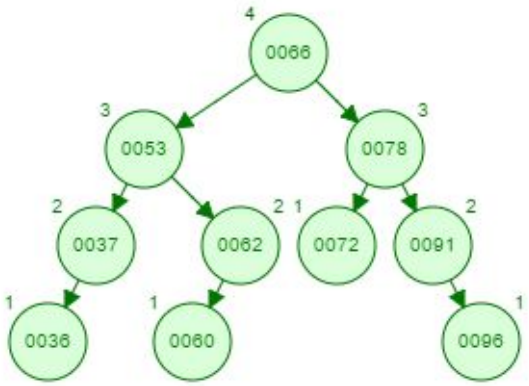
(do while-u by sa dala dať negovaná posledná podmienka)

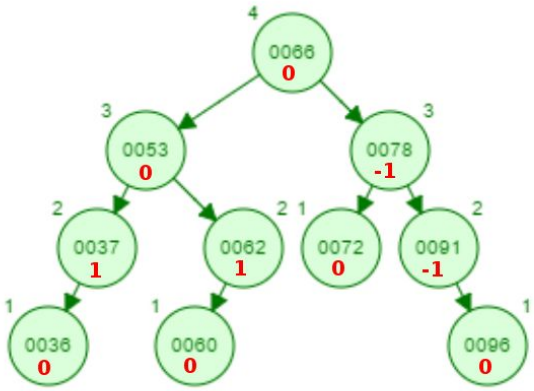
E

pomôcka na všetky stromy: http://people.ksp.sk/~kuko/gnarley-trees/

// tie cisla pri uzloch ale zrejme nie su hodnoty faktoru vyvazenia ale vyska (?) uzla… ci?

// faktor vyvazenia by mal byt vyska laveho - vyska praveho podstromu (videl som aj opacnu verziu, takze asi zalezi od implementacie); spravne by asi malo byt to v 2. verzii (faktory su cervenou)



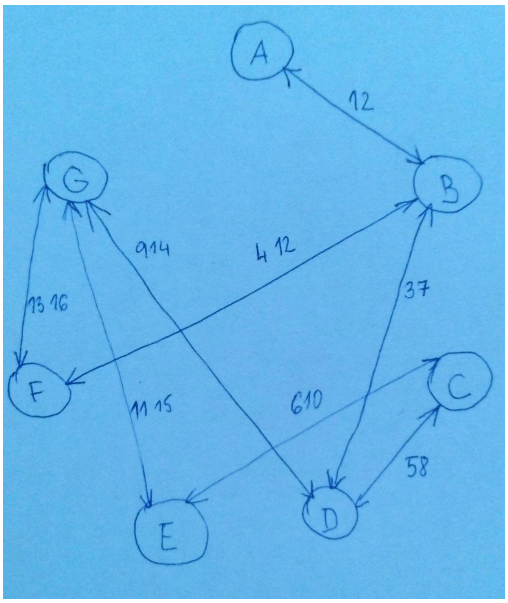


F

vyriešené na papieri

(zaujímavé, že niekde v prednáške je aj matica s hodnotami uvedená ako matica susednosti a nie binárna matica)

G



Čísla sú náhodné (nepodstatné), dôležité sú šípky.

H

https://www.cs.usfca.edu/~galles/visualization/RadixSort.html

	346	22	31	212	157	102	568	435	8	14	5
po jednotkách:	31	22	212	102	14	435	5	346	157	568	8
po desiatkách:	102	5	8	212	14	22	31	435	346	157	568
po stovkách:	5	8	14	22	31	102	157	212	346	435	568

// To pomocne pole by malo byt o vekosti 11? a pole[0] zacina od 1 alebo 0 //prečo 11? ak myslíš pomocné pole s ciframi tak tie sú od 0 do 9, čo je dokopy 10 čísel..

I

rovnaké ako 2014

Hashovacia tabuľka

Postupne prechádzať čísla v poli a pozrieť sa do tabuľky (na začiatku je prázdna), či sa tam nenachádza s-n[i], ak nie, tak uložiť číslo do tabuľky a pokračovať ďalším číslom. Ak áno, tak sa číslo našlo, keď sme prešli celé pole, tak sa číslo nenašlo. **// vie toto niekto vysvetliť? veď tam je že súčet dvoch čísel a nie, že či číslo…**
// pre každý prvok poľa A[i] a pre hľadaný súčet s sa pozrieš, či v heš tabuľke sa už nachádza s - A[i]. Ak áno, tak to znamená, že daný súčet je možné vytvoriť a skončí. Ak nie, vloží do heš tabuľky A[i] a ide ďalej

J

Kruskal:	4	9	18	19	21	22	23	30
Prim:	9	4	22	30	19	21	23	18

K

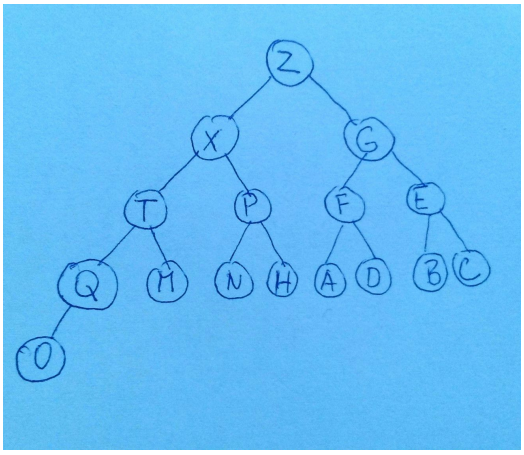
rovnaké ako 2014

hrubou sú čiary vyznačené na papieri

vrchol:	A	C	D	F	H	E	B	G	I
vzdialenosť:	0	1	12	20	25	28	34	40	53

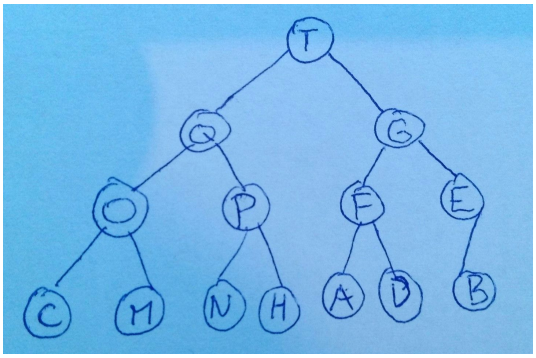
L

a)



Z sa pridá ako ľavé dieťa písmena O. Z sa porovná s rodičom O, pričom sa zistí že rodič je menší a tak sa vymenia. Potom sa Z porovná s rodičom Q, zistí sa, že Z je väčšie a vymenia sa … takto to pokračuje, pokým Z nieje menšie ako rodič, alebo Z sa prebublá nahor(celková postupnosť Z>O, Z>Q, Z>T a Z>X), až sa stane koreň (tento prípad).+

b)



Má sa vymazať X. X sa vymení s C a vymaže sa. C je koreň a (v maxheape) sa porovná s deckom, ktoré je väčšie, v tomto prípade to bude T>G , potom sa

takto prebubláva C(výmena T s C), až pokým je väčšie ako obidve deti, alebo je už celkom dole (tento prípad). Po T>G sa porovnávajú Q>P a O>M.

M

B-2
(pomôcka: posledné dva prvky sú usporiadané ako vo výslednej postupnosti, čiže dva)

N

rovnaké ako 2014
phase2: C->D X 0 (do D)
phase3: D->B X -1 (do B)
ostatne riadky su prazdne alebo niesu succesful relax

ďalšia príprava:

http://tomas-haber.blogspot.sk/2009/02/skuska-z-predmetu-dsa-opravny-termin.htm
https://github.com/citruslee/Studijne-materially-FIIT/blob/master/3.%20Semester/DSA/skusky/DSA-vypracovaneOT.pdf
http://www.cs.princeton.edu/courses/archive/fall13/cos226/exams.php
https://www.fiitkar.sk/wiki/index.php/D%C3%A1tov%C3%A9_%C5%A1trukt%C3%BAry_a_algoritmy_sk%C3%BA%C5%A1ky#Opravn.C3.BD_term.C3.ADn

Opravák 2015:

udrziavanie medianu (z min ulych rokov) -
http://stackoverflow.com/questions/15319561/how-to-implement-a-median-heap (8)
djikstra - napisat postupnost vrcholov a hodnoty, z toho isteho grafu urobit prima z ktorehokolvek vrcholu a nakreslit kostru (5)
djikstra - pocet najkratsich ciest , ako z riadneho 2014, napisat upravene funkcie
doplnenie prvku do avl a faktory (3)
zoradovanie boolean poľa v čase O(n) a pamatou O(1) (4)
nakodit Quickselect (4)
linearne skusanie (3)
vybrat postupnost ktore je nejakou haldou a potom doplnit prvok do nej a nakreslit ju ako strom (3)
hladanie k-teho prvku v halde, cislo prvku je take ako pri postupnosti pri hladani inorder

Dátové štruktúry a algoritmy

štvrtok, 10. januára 2013

Podpíšte tento list aj dvojhárok – meno, priezvisko a osobné číslo.
Odpovede píšte priamo na tento list všade, kde sa to dá (máte to aj naznačené). Pracujete samostatne a odovzdáte len výsledky vlastnej práce, dosiahnuté bez pomoci.
Meno a priezvisko: *osobné číslo:*

A	B	C	D	E	F	G	H	I	J	bonus	
3	3	4	4	5	4	2	6	4	10	10	
3	3	2	0	5	4	2	2	8	/	0	29

A 3 Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je ~~7~~ 9 (miest pamäti) a rozptylová funkcia je h(x) = i(x) mod 9, kde i(x) je poradové číslo písmena x v (anglickej) abecede, pričom kolizie sa riešia lineárnym skúšaním. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) H O L U B I C A. *(Nekreslite priebežné stavy.)*

x	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
i(x)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

riešenie:

0	1	2	3	4	5	6	7	8	9
I	A	B	L	U	C	O		H	

3

11

B 3 Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je 8 (miest pamäti) a rozptylová funkcia je h(x) = x mod 11, pričom kolízie sa riešia a) zreťazením b) lineárnym skúšaním. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 8,20,29,16,21,18,11,34.

a)

0	1	2	3	4	5	6	7	8	9	10
11	34				16	6	29		20	21

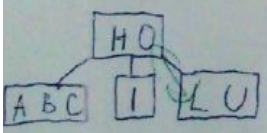
18

b)

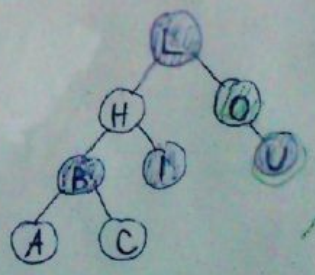
0	1	2	3	4	5	6	7	8	9	10
11	34				16	6	29	18	20	21

3

C 4 Nakreslite 2-3-4 strom, ktorý vznikne postupným vkladaním kľúčov
H O L U B I C A
do stromu, ktorý bol na začiatku prázdny.
Nakreslite červeno-čierny strom, ktorý vznikne postupným vkladaním tých istých kľúčov do stromu, ktorý bol na začiatku prázdny.



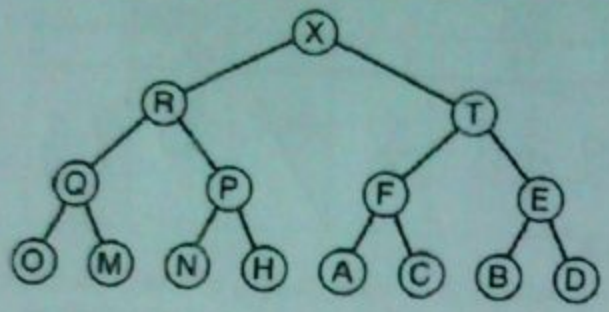
2,3,4 strom



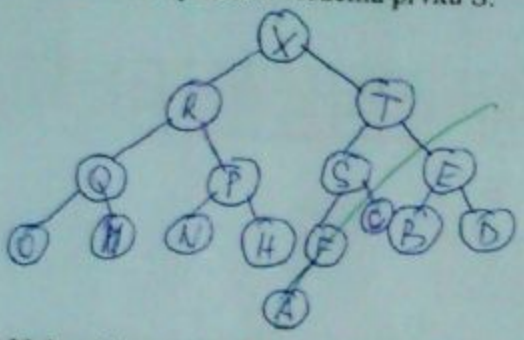
Č-Č strom

vrchol O - čierny
vrchol U - červený

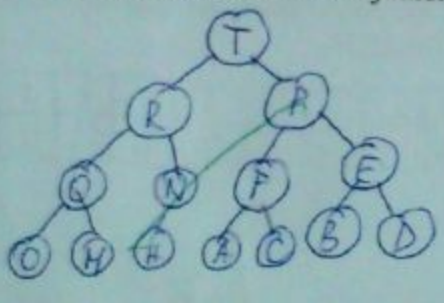
D 4 Uvažujte túto max-haldu (relácia usporiadania je daná abecedou).



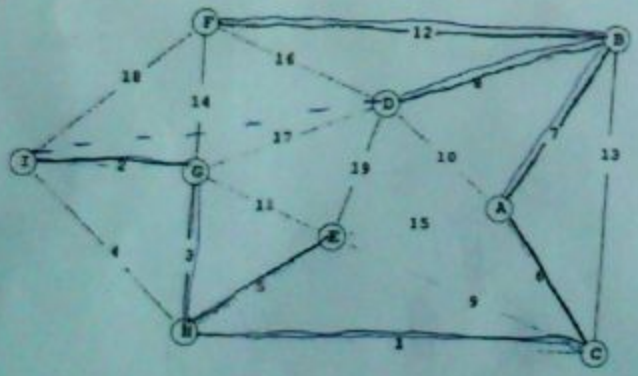
a) Nakreslite výsledok vloženia prvku S.



b) Nakreslite výsledok zrušenia najväčšieho prvku z pôvodnej haldy (t.j. pred vložením S).



E 5 Uvažujte tento hranovo ohodnotený neorientovaný graf s 9 vrcholmi a 19 hranami. Všimnite si, že váhy hrán sú rôzne celé čísla od 1 do 19.



a) Napište postupnosť hrán, ktoré zaradí Kruskalov algoritmus do minimálnej kostry grafu, v poradí, v akom ich zaraďuje.

1 2 3 5 6 7 8 12   2

b) Predpokladajte, že hrana D-I s váhou w sa pridá do grafu. Čo musí platiť o w, aby sa hrana D-I dostala do minimálnej kostry grafu?

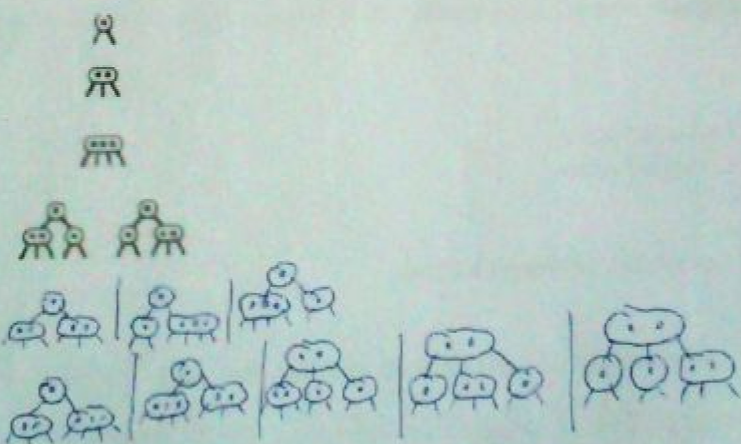
- mala by byť menšia ako váha hrany B-D, čiže w < 8   1

c) Napište postupnosť hrán, ktoré zaradí Primov algoritmus do minimálnej kostry grafu, v poradí, v akom ich zaraďuje. Algoritmus začína vo vrchole A.

6 1 3 2 5 7 8 12   2

F 4 Nizsie je naznačená tabuľka všetkých možných tvarov 2-3-4 stromov, ktoré môžu vzniknuť vložením N rôznych kľúčov do stromu, ktorý bol na začiatku prázdny, pre N od 1 po 6. V ľavom stĺpci je počet kľúčov, v ďalšom stĺpci je počet možných rôznych tvarov stromov s toľkých kľúčmi a ďalej vpravo sú nakreslené tie tvary.
Doplňte posledné dva riadky tabuľky (nakreslite 3 tvary stromov s 5 kľúčmi a napište počet rôznych tvarov stromov s 6 kľúčmi a nakreslite ich),

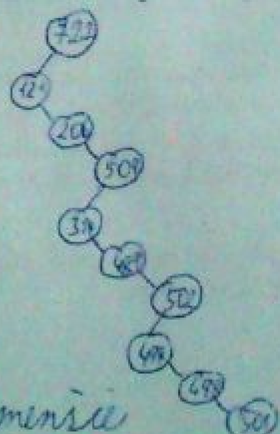
1	1
2	1
3	1
4	2
5	3
6	5



4

G 2 Majme binárny vyhľadávací strom s číslami z intervalu [1, 1000]. Chceme nájsť číslo 501. Môže byť táto postupnosť postupnosťou, v akej sa postupne pri hľadaní navštevujú vrcholy? Prečo áno alebo prečo nie? (Pri zdôvodnení si pomôžte obrázkom.)

722, 121, 206, 509, 314, 489, 502, 494, 498, 501.



- áno môže, navštívené vrcholy spĺňajú podmienku BVS ~~(vľavo~~ ~~menšie čísla vľavo a v pravo~~ väčšie čísla, a každý vrchol) -> ľavý podstrom vrchola obsahuje menšie čísla a pravý podstrom väčšie

2

H 6 bodov Napíšte algoritmus, ktorý usporiada postupnosť celých čísiel z intervalu (0, 1000) zapísaných v A[1..n] v postupnom poradí. Napíšte asymptotický odhad časovej zložitosti Vášho algoritmu vyjadrený pomocou počtu operácií sprístupnenia prvku poľa alebo pomocou nejakej inej realistickej metriky. Čím rýchlejší algorimus napíšete, tým bude zaň viac bodov.

Riešenie môžete písať na dvojhárok, označte „sort."

I 10 Dynamické určovanie mediánu. Načítava sa veľký počet čísiel, ktoré sú všetky navzájom rôzne. Treba stále udržiavať hodnotu mediánu už prečítanej množiny čísiel. Napr. po prečítaní čísiel 2 9 7 4 1 treba vrátiť 4. Keď sa ďalej prečítajú čísla 6 8 5 a vyžiada sa medián, treba vrátiť 5 alebo 6. Pri návrhovaní riešenia treba dodržať tieto požiadavky:

Možno použiť pomocnú pamäť iba v konštantnom rozsahu (navyše toho, čo treba pre zapísanie samotných čísiel).
Na vrátenie mediánu máte iba konštantný čas.
Na spracovanie n-tého prvku máte čas najviac úmerný log n.

Možno zmazať časť už prečítaných čísiel tak, aby sa neskôr vždy dal určiť medián všetkých prečítaných čísiel?

ano
nie

(zakrúžkujte)

Predpokladajme, že sa načítalo n čísiel a ich medián je v. Ktoré z týchto tvrdení platia o medián po spracovaní n+1. prvku?

A. Nezmenil sa.
B. Je to najväčšie z čísiel menších ako v.
C. Je to najmenšie z čísiel väčších ako v.
D. Buď A. alebo B. alebo C.
E. Buď B. alebo C, ale nie A.
F. Môže to byť ľubovoľné z doteraz prečítaných čísiel.

(zakrúžkujte)

Jednou či dvoma vetami opíšte, ako by ste vyriešili tento problém. (naozaj 2 vety, nie viac!)

dve vety: Prečítané čísla by som udržiaval v binárnej halde.

Bonus 10 Navrhnite algoritmus na nájdenie najčastejšie sa vyskytujúceho prvku (symbolu alebo celého čísla) v danej postupnosti dĺžky n:

a) prvky sú symboly z nejakej usporiadanej množiny
b) prvky sú celé čísla z nejakého intervalu [0, q].

Algoritmus musí byť čo najrýchlejší. Urobte odhad jeho časovej zložitosti a odôvodnite ho. Algoritmus opíšte v algoritmickom jazyku.

Podpíšte tento list aj dvojhárok – meno, priezvisko a osobné číslo.
Odpovede píšte priamo na tento list. Pracujete samostatne a odovzdáte len výsledky vlastnej práce, dosiahnuté bez pomoci.
Meno a priezvisko: *osobné číslo:*

A	B	C	D	E	F	G	H	I	J	K	bonus	
3	3	2	2	2	2	8	2	7	6	8	10	

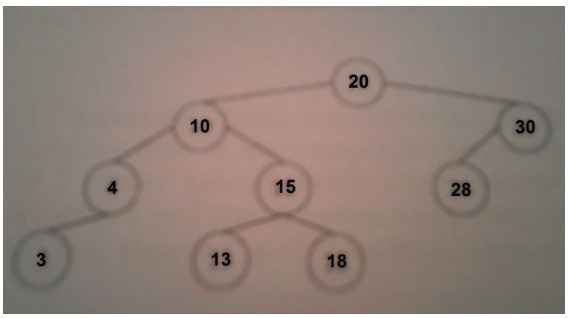
A 3 Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je 6 (miest pamäti) a rozptylová funkcia je h(x) = x mod 6, pričom kolízie sa riešia zreťazením. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 35, 2, 18, 6, 3, 10, 8, 5.
(Nekreslite priebežné stavy.)

B 3 Uvažujte prázdnu rozptylovú tebuľku, ktorej veľkosť je 11 (miest pamäti) a rozptylová funkcia je h(x) = x mod 11, pričom kolízie sa riešia lineárnym skúšaním. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 35, 2, [illegible], 6, 3, 10, 8, 5.

1

C 2 Uvažujte AVL strom na obrázku. Nakreslite strom po vložení prvku 17.

1



D 2 Toto je pole práve po prvom rozčlenení v algoritme rýchleho usporadúvania (quicksort):
3, 0, 2, 4, 5, 8, 7, 6, 9
Ktorý z týchto prvkov mohol byť pivot? (môže ich byť viac takých!)

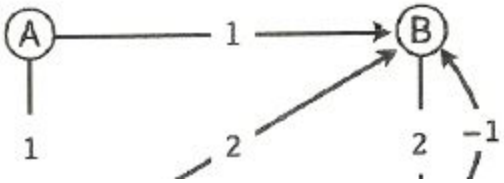
E 2 Predpokladajme, že sa usporadúva pole siedmich celých čísiel (pomocou haldy, heapsort). Stav po jednom z vyhaldovaní (heapify) je:
6 4 5 1 2 7 8
Koľko vyhaldovaní už prebehlo?

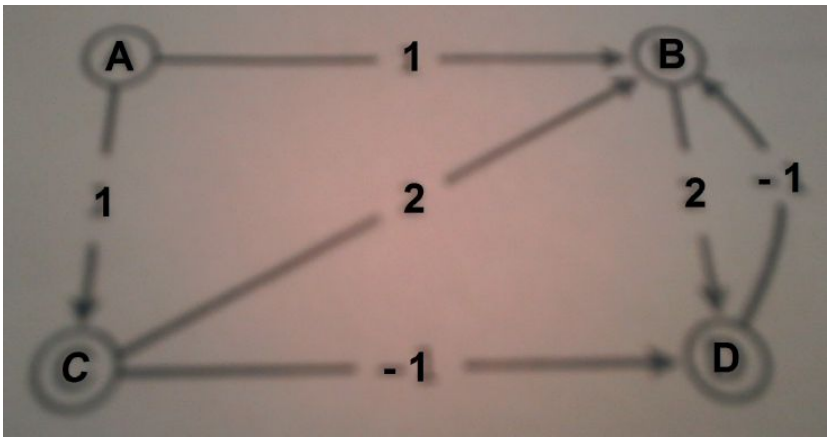
A. 1
B. 2
C. 3 alebo 4
D. 5 alebo 6

F 2 Predpokladajme, že pri usporadúvaní 100 prvkov výberom (selection sort) práve prebehlo 42 opakovaní hlavného cyklu. Koľko prvkov je zaručene na svojom definitívnom mieste (a už sa s nimi určite nebude hýbať)?

A. 21
B. 41
C. 42
D. 43

G 8 Bellman-Ford Uvažujte orientovaný graf s ohodnotenými hranami, ktorý má aj záporne ohodnotené hrany, avšak nemá záporné cykly. Dokončite stopu výpočtu Bellmanovho-Fordovho algoritmu, ktorý sa začne vykonávať vo vrchole A, ako je znázornená v tabuľke nižšie. Ako pomôcka je uvedená stopa prvej fázy relaxácie. (successful relax = úspešná relaxácia hrany, shortest known distance to = najkratšia doteraz známa vzdialenosť do). Hrany sa v každej fáze relaxujú v poradí, v akom sú uvedené pod sebou v tabuľke.





		successful relax?	shortest known distance to			
			A	B	C	D
			0			
phase 1	D→B					
	C→D					
	C→B					
	B→D					
	A→C	X			1	
	A→B	X		1		
phase 2	D→B					
	C→D					
	C→B					
	B→D					
	A→C					
	A→B					
phase 3	D→B					
	C→D					
	C→B					
	B→D					
	A→C					
	A→B					
phase 4	D→B					
	C→D					
	C→B					
	B→D					
	A→C					
	A→B					

H 2 Knuth Morris Pratt. KMP algoritmus na hľadanie výskytu vzorového reťazca p v texte využíva predvypočítanú predponovú funkciu, ktorá je daná tabuľkou. Pre vzor p=ababaca ukážte zvyšné dva kroky počitania predponovej funkcie (doplňte hodnoty do tabuľky).

q	1	2	3	4	5	6	7
p	a	b	a	b	a	c	a
π	0	0	1	2	3		

I 7 Sedem algoritmov usporadúvania. Nižšie je znázornený vľavo stĺpec so vstupnou postupnosťou reťazcov, ktoré sa majú usporiadať, vpravo stĺpec s výslednou usporiadanou postupnosťou. Medzi nimi je 7 stĺpcov znázorňujúcich nejaký priebežný stav usporadúvania podľa niektorého z algoritmov, uvedených pod tým aj s písmenami, ktoré ich pre jednoduchosť označujú. Napíšte pod každý stĺpec písmeno algoritmu, podľa ktorého sa v tom stĺpci usporadúva.

fuzz	zoom	doze	benz	cozy	cozy	benz	cozy	benz
cozy	zest	cozy	cozy	czar	czar	buzz	fuzz	buzz
zinc	zone	czar	zinc	doze	fuzz	cozy	quiz	cozy
quiz	ritz	benz	quiz	fuzz	hazy	cruz	zinc	cruz
zero	zero	faze	zero	gaze	laze	czar	hazy	czar
suez	suez	cruz	suez	hazy	maze	daze	suez	daze
zone	zing	haze	zone	jazz	ouzo	doze	zero	doze
hazy	quiz	buzz	hazy	laze	quiz	faze	zone	faze
maze	maze	fuzz	maze	maze	suez	fuzz	czar	fuzz
ouzo	ouzo	gaze	ouzo	ouzo	zero	gaze	laze	gaze
czar	zeal	ritz	czar	quiz	zinc	haze	maze	haze
laze	raze	daze	laze	suez	zone	hazy	ouzo	hazy
doze	lazy	maze	doze	zero	doze	zone	doze	jazz
gaze	zeta	lazy	gaze	zinc	gaze	ouzo	gaze	laze
zing	zinc	whiz	zing	zing	zing	zing	jazz	lazy
jazz	jazz	hazy	jazz	zone	jazz	jazz	zing	maze
zoom	hazy	ooze	zoom	benz	zoom	zoom	buzz	ooze
cruz	cruz	ouzo	cruz	buzz	cruz	quiz	cruz	ouzo
ritz	cozy	zeal	ritz	cruz	ritz	ritz	ritz	quiz
buzz	buzz	jazz	buzz	daze	buzz	zinc	zoom	raze
faze	faze	zero	faze	faze	faze	laze	faze	ritz
zest	czar	size	zest	haze	zest	zest	raze	size
zeal	fuzz	zinc	zeal	lazy	zeal	zeal	zeal	suez
raze	laze	laze	raze	ooze	raze	raze	zest	whiz
ooze	ooze	zeta	ooze	raze	ooze	ooze	daze	zeal
lazy	doze	suez	lazy	ritz	lazy	lazy	haze	zero
haze	haze	zing	haze	size	haze	zero	lazy	zest
daze	daze	quiz	daze	whiz	daze	suez	ooze	zeta
zeta	gaze	zoom	zeta	zeal	zeta	zeta	benz	zinc
size	size	zest	size	zest	size	size	size	zing
whiz	whiz	zone	whiz	zeta	whiz	whiz	whiz	zone
benz	benz	raze	fuzz	zoom	benz	maze	zeta	zoom

B	F	D	G	C	E	A

A. nerekurzívne usporadúvanie zlučovaním (Bottom-up mergesort)
B. usporadúvanie haldovaním (Heapsort)
C. usporadúvanie vkladaním (Insertion sort)
D. rýchle usporadúvanie (Quicksort)
E. usporadúvanie výberom (Selection sort)
F. Shellovo usporadúvanie (Shellsort)
G. rekurzívne usporadúvanie zlučovaním (Top-down mergesort)

J 6 Bitónne maximum

Postupnosť $x_0, \ldots, x_{n-1}$ je bitónna práve vtedy, ak pre nejaké x_k, k □ {0, . . . , n − 1} platí, že všetky prvky pred ním (vrátane jeho samotného) tvoria rastúcu postupnosť, kdežto prvky stojace za ním tvoria klesajúcu postupnosť. Formálne zapísané musí platiť, že:
$x_0 \leq x_1 \leq \ldots \leq x_k \geq x_{k+1} \geq \ldots \geq x_{n-1}$. Navrhnite algoritmus, ktorý určí najväčší prvok v bitónnej postupnosti s N prvkami zapísanej v jednorozmernom poli v čase úmernom log N.

Dátové štruktúry a algoritmy streda, 11. januára 2012

Podpíšte tento list aj dvojhárok – meno, priezvisko a osobné číslo.
Odpovede píšte priamo na tento list. Pracujete samostatne a odovzdáte len výsledky vlastnej práce, dosiahnuté bez pomoci.
Meno a priezvisko: *osobné číslo:*

(a) Opíšte algoritmus, použite stručný jasný opis najlepšie v algoritmickom jazyku, nepíšte úplný kód v programovacom jazyku. Hodnotiť sa bude správnosť, efektívnosť a čistota algoritmu.

(b) Ukážte, ako pracuje Váš algoritmus uvedením prvých štyroch porovnaní, ktoré by vykonal pri hľadaní maxima v tomto 15-prvkovom bitónnom poli:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a[i]	10	34	56	76	87	80	70	66	56	30	28	25	20	15	11

1.
2.
3.

K 8 Dynamické určovanie mediánu. Načítava sa veľký počet čísiel, ktoré sú všetky navzájom rôzne. Treba stále udržiavať hodnotu mediánu už prečítanej množiny čísiel. Napr. po prečítaní čísiel 2 9 7 4 1 treba vrátiť 4. Keď sa ďalej prečítajú čísla 6 8 5 a vyžiada sa medián, treba vrátiť 5 alebo 6. Pri navrhovaní riešenia treba dodržať tieto požiadavky:

- Možno použiť pomocnú pamäť iba v konštantnom rozsahu (navyše toho, čo treba pre zapísanie samotných čísiel).
- Na vrátenie mediánu máte iba konštantný čas.
- Na spracovanie n-tého prvku máte čas, najviac úmerný log n.

Možno zmazať časť už prečítaných čísiel tak, aby sa neskôr vždy dal určiť medián všetkých prečítaných čísiel?

a) áno
b) nie

Predpokladajme, že sa načítalo n čísiel a ich medián je v. Ktoré z týchto tvrdení platia o mediáne po spracovaní n+1. prvku?

Dátové štruktúry a algoritmy streda, 11. januára 2012

Podpíšte tento list aj dvojhárok – meno, priezvisko a osobné číslo.
Odpovede píšte priamo na tento list. Pracujete samostatne a odovzdáte len výsledky vlastnej práce, dosiahnuté bez pomoci.
Meno a priezvisko: *osobné číslo:*

A. Nezmenil sa.
B. Je to najväčšie z čísiel menších ako v.
C. Je to najmenšie z čísiel väčších ako v.
D. Buď A. alebo B. alebo C.
E. Buď B. alebo C, ale nie A.
F. Môže to byť ľubovoľné z doteraz prečítaných čísiel.

Jednou či dvoma vetami opíšte, ako by ste vyriešili tento problém. (naozaj 2 vety, nie viac!)

Bonus 10
Nech je dané pole N 64-bitových dlhých celých čísiel so znamienkom. Treba nájsť dve čísla x a y také, že x+y=0. (Pre jednoduchosť predpokladajte, že ani jedno z čísiel nie je 0 alebo -2^{63}).
Opíšte efektívny algoritmus v rozsahu najviac pol strany. Váš algoritmus by mal počítač v lineárnom čase v priemernom prípade prípadne aspoň čase N log N (za menej bodov).
Vaša odpoveď sa bude hodnotiť podľa správnosti, jasnosti a úspornosti vyjadrenia.

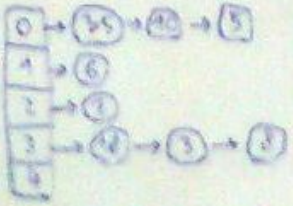
Zakrúžkujte odhad času vykonania Vášho algoritmu v priemernom prípade:
logN N N logN N^2 2^N

Zakrúžkujte odhad času vykonania Vášho algoritmu v najhoršom prípade:
logN N N logN N^2 2^N

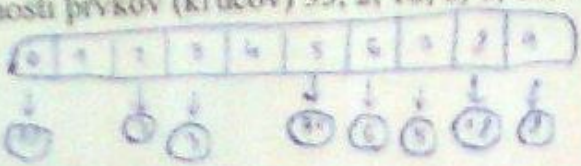
Dátové štruktúry a algoritmy utorok, 11. januára 2011

Podpíšte tento list aj dvojhárok – meno, priezvisko a osobné číslo.
Odpovede píšte priamo na tento list. Pracujete samostatne a odovzdáte len výsledky vlastnej práce, dosiahnuté bez pomoci.
Meno a priezvisko: *osobné číslo:*

1 – (3b) – Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je 5 (miest pamäti) a rozptylová funkcia je h(x) = x mod 5, pričom kolízie sa riešia zreťazením. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 35, 2, 18, 6, 3, 10, 8, 5. (Nekreslite priebežné stavy.)

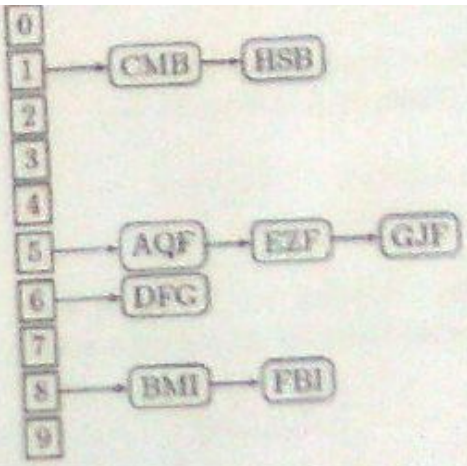


2 – (3b) – Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je 10 (miest pamäti) a rozptylová funkcia je h(x) = x mod 10, pričom kolízie sa riešia lineárnym skúšaním. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 35, 2, 18, 6, 3, 10, 8, 5. (Nekreslite priebežné stavy.)



3 – (3b) – Uvažujte rozptylovú funkciu s riešením kolízií reťazením.

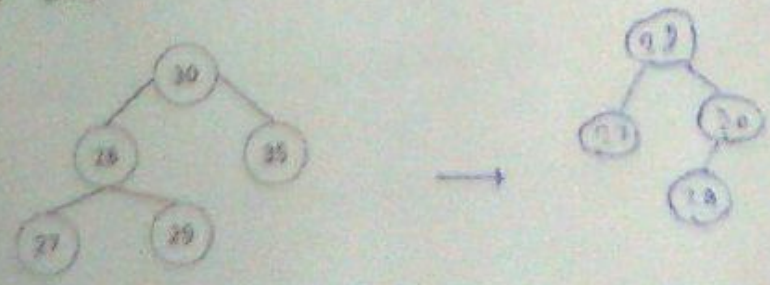
Ukážte, ako by vyzerala rozptylová tabuľka (na obrázku) s tými istými prvkami, ak by sa použilo na riešenie kolízií otvorené adresovanie s lineárnym skúšaním.



Je Vaše riešenie predchádzajúcej otázky jedinečné? Ak áno, napíšte prečo. Ak nie, napíšte prečo a nakreslite príklad iného riešenia.

po vymazaní prvku 35

4 – (2b) – Uvažujte AVL strom na obrázku nižšie. Nakreslite strom po vymazaní prvku 35.



5 – (6b) – Napíšte algoritmus (môže byť v pseudokóde), ktorý vzostupne usporiada postupnosť celých čísiel z intervalu (0, 1000) zapísaných v A[1..n]. Napíšte asymptotický odhad časovej zložitosti Vášho algoritmu vyjadrený pomocou počtu operácií sprístupnenia prvku poľa alebo pomocou nejakej inej realistickej metriky. Čím rýchlejší algoritmus napíšete, tým bude zaň viac bodov.

(odpoveď píšte na dvojhárok)

6 – (4b) – Napíšte algoritmus (môže byť v pseudokóde), ktorý z danej binárnej min-haldy nájde (vypíše) všetky hodnoty menšie než x.

(odpoveď píšte na dvojhárok)

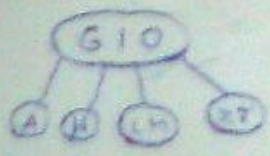
7 – (2b) – Nech Σ^* označuje množinu všetkých možných reťazcov nad abecedou Σ. Čo znamená, že w je prípona reťazca x? *(zakrúžkujte vždy áno alebo nie)*

Pre nejaké $y \in \Sigma^*$, $wy=x$	áno / (nie)
Pre nejaké $y \in \Sigma^*$, $yw=x$	(áno) / nie
Pre nejaké $y \in \Sigma$, $wy=x$	áno / (nie)
Pre nejaké $y \in \Sigma$, $yw=x$	(áno) / nie
Pre nejaké $y \in \Sigma^*$, wyx	áno / (nie)

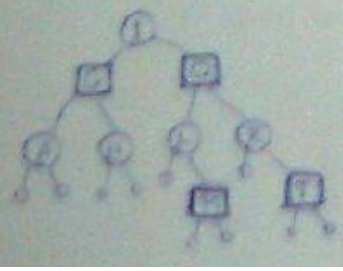
8 – (2b) – Algoritmus Boyera a Moora na porovnávanie reťazcov porovnáva vzor s textom *(zakrúžkujte vždy áno alebo nie)*

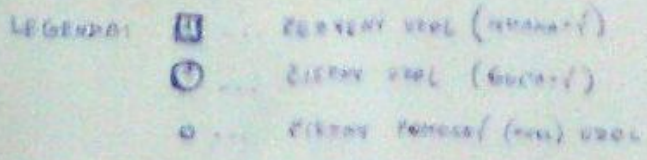
Zľava doprava	(áno) / nie	Zhora nadol	(áno) / nie
Sprava doľava	áno / (nie)	Zdola nahor	áno / (nie)

9 – (4b) – Nakreslite 2-3-4 strom, ktorý vznikne postupným vkladaním kľúčov A, L, G, O, R, I, T, H, M do stromu, ktorý bol na začiatku prázdny.



Nakreslite červeno-čierny strom, ktorý vznikne postupným vkladaním tých istých kľúčov do stromu, ktorý bol na začiatku prázdny.





10 – (6b) – 6 algoritmov usporadúvania. V najľavejšom stĺpci je vstupná postupnosť reťazcov, ktorá sa má usporiadať. V najpravejšom stĺpci je výsledná usporiadaná postupnosť. V ostatných stĺpcoch medzi nimi sú nejaké priebežné stavy usporadúvania jedným z 6 algoritmov uvedených nižšie. Pod každý z týchto stĺpcov napíšte písmeno príslušného algoritmu. Každé písmeno použite práve raz.

that	been	also	also	into	been	year	also
even	even	down	back	even	even	with	back
than	ever	come	been	than	from	will	been
been	fell	been	come	been	more	more	come
from	from	back	down	from	next	were	down
next	loss	even	even	next	over	plea	even
show	more	next	even	show	plea	well	ever
with	next	into	fell	jobs	show	lost	fell
more	over	fell	from	more	than	even	from
were	plea	from	have	much	that	some	have
over	show	jobs	into	over	were	very	into
plea	than	next	jobs	plea	with	next	jobs
fell	that	have	with	fell	fell	lead	lead
time	time	lead	time	back	time	time	loss
loss	were	loss	loss	loss	loss	that	lost
ever	with	over	show	ever	ever	jobs	more
lost	also	lost	lost	lost	lost	been	much
also	come	more	that	also	also	also	next
down	down	much	more	down	down	down	over
said	have	show	said	said	said	said	plea
some	lost	plea	some	some	some	from	said
have	said	that	were	have	have	have	show
very	some	said	very	lead	very	over	some
come	very	some	than	come	come	come	than
into	back	will	over	that	into	into	that
lead	into	very	lead	very	lead	fell	time
back	jobs	time	next	time	back	back	very
year	lead	than	year	year	year	than	well
will	much	with	will	will	will	show	were
well	well	well	well	well	well	loss	will
much	will	were	much	were	much	much	with
jobs	year	year	plea	with	jobs	ever	year

E F C A B D

A. Shellovo usporadúvanie
B. Usporadúvanie vkladaním
C. Rýchle usporadúvanie (bez znáhodnenia)
D. Usporadúvanie výberom
E. Usporadúvanie zlučovaním
F. Heapsort

F. Heapsort

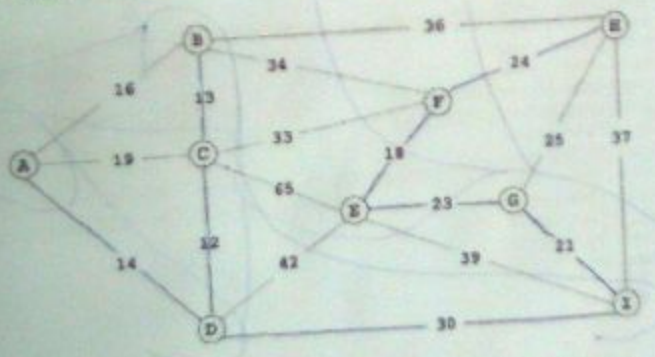
11 – (4b) – Na obrázku (na nasledujúcej strane) je naznačená tabuľka všetkých možných tvarov 2-3-4 stromov, ktoré môžu vzniknúť vložením N rôznych kľúčov do stromu, ktorý bol na začiatku prázdny, pre N od 1 po 6. V ľavom stĺpci je počet kľúčov, v ďalšom stĺpci je počet možných rôznych tvarov stromov s toľkých kľúčmi a ďalej vpravo sú nakreslené tvary týchto stromov.

3 / 5

Doplňte posledné dva riadky tabuľky (nakreslite všetky 3 možné tvary stromov s 5 kľúčmi a napíšte počet rôznych tvarov stromov s 6 kľúčmi a nakreslite ich).

1	1	
2	1	
3	1	
4	2	
5	3	
6	5	

12 – (6b) – Uvažujte hranovo ohodnotený neorientovaný graf:



Napíšte zoznam hrán v minimálnej kostre tohto grafu, ako ich bude do nej pridávať Kruskalov algoritmus. (Pomôcka: váhy hrán v stúpajúcom poradí sú: 12 13 14 16 18 19 21 23 24 25 30 33 34 36 37 39 42 65)

Napíšte zoznam hrán v minimálnej kostre tohto grafu, ako ich bude do nej pridávať Primov algoritmus, ktorý začne od vrchola A.

Bonus – (10b)

Uvažujme usporiadanú množinu prvkov S. Binárna min-max halda T prvkov S je dátová štruktúra (binárny strom) s vlastnosťami ako bežná binárna halda avšak:

- T je min-max usporiadaná, t.j. hodnoty zapísané vo vrcholoch na párnej (nepárnej) úrovni sú menšie (väčšie) alebo rovné než hodnoty zapísané v ich nasledovníkoch (ak nejakých majú). Koreň je na úrovni 0. Takže najmenšia hodnota z množiny S je zapísaná v koreni T a najväčšia hodnota z množiny S je zapísaná v jednom z nasledovníkov koreňa.
- Min-max halda s n prvkami sa dá zapísať do vektora A[1..n].

Implementujte operáciu vloženia prvku do min-max haldy.