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)

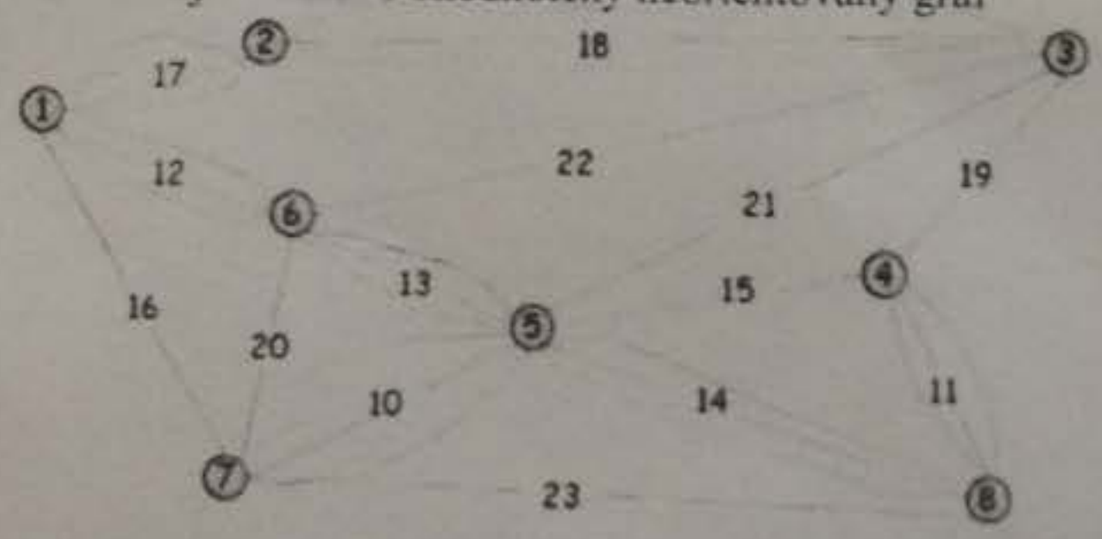
```
1  for j ← 2 to length[A]
2        do key ← A[j]
3           i ← j − 1
4           while i > 0 and A[i] ≥ key
5                 do A[i + 1] ← A[i]
6                    i ← i − 1
7           A[i + 1] ← key
```

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.

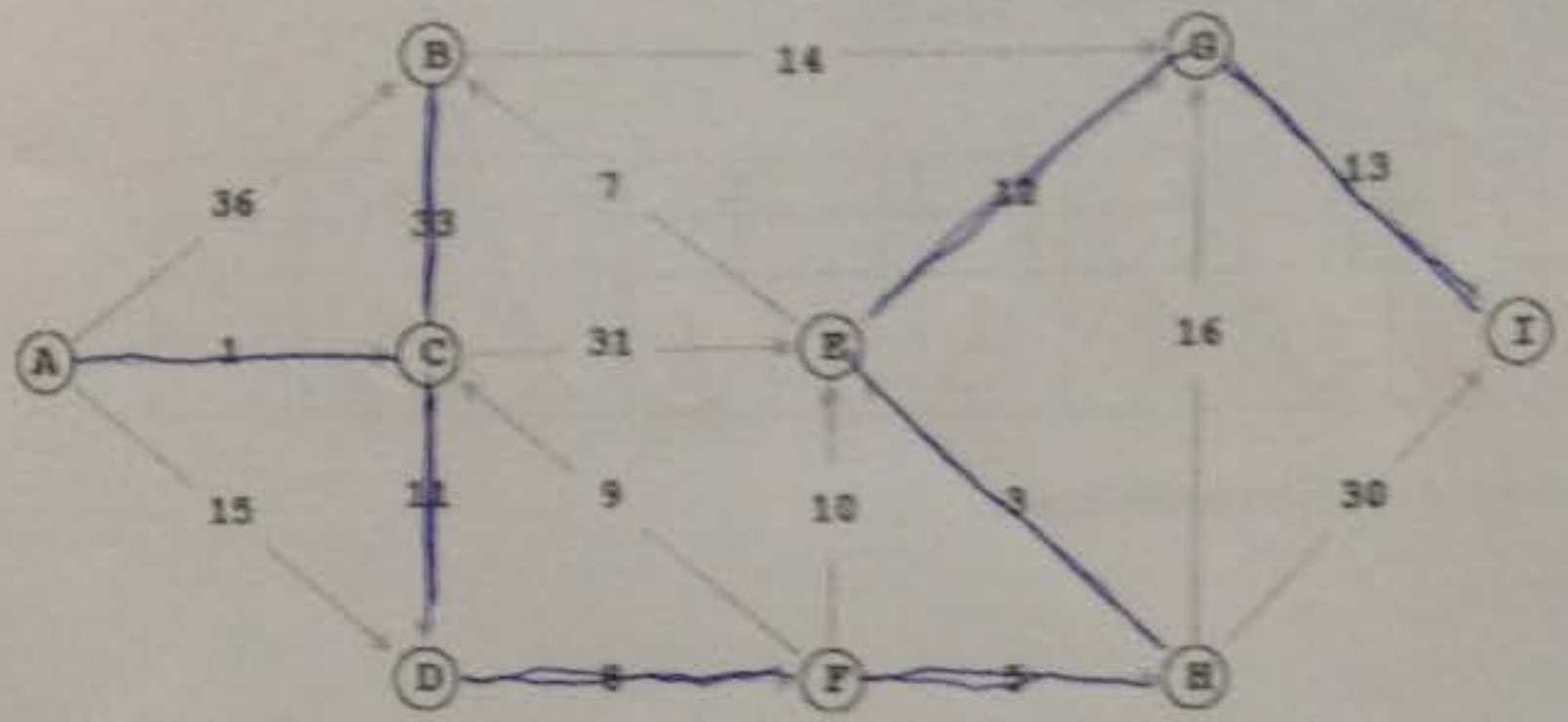
C 6 Daná je postupnosť n čísel a jedno číslo, ktoré nazveme s. Navrhnite algoritmus, ktorý rozhodne v lineárnom čase, či súčet niektorých dvoch čísel v postupnosti je s. Nemusíte písať presne v nejakom programovacom jazyku, stačí pseudokód alebo slovný opis.

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 medzí (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.

4A. 0 2 x y z
3B. 3 4 v w
C. 0 4 v w x y z
D. 5 8 r s t u
E. 0 8 r s t u v w x y z
1F. 0 1 y z
7G. 7 8 r s
2H. 5 6 t u

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

G	A	B	F				C	E
				A		D		

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
-	I	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		

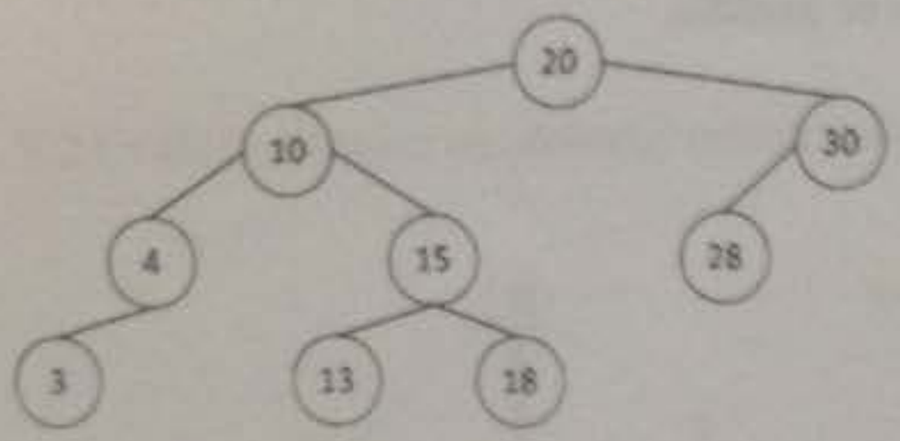
H 6 Hľadanie 2D vzoru. Štandardné algoritmy na vyhľadávanie reťazcov (KMP, RK, BM), fungujú pre 1D vzory (lineárny reťazec znakov). 2D vzor nech je matica P1 riadkov a P2 stĺpcov znakov. Vzor vyhľadávame v 2D mape (matici znakov), ktorá má M1 riadkov a M2 stĺpcov. Navrhnite algoritmus – nejakú kombináciu 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.

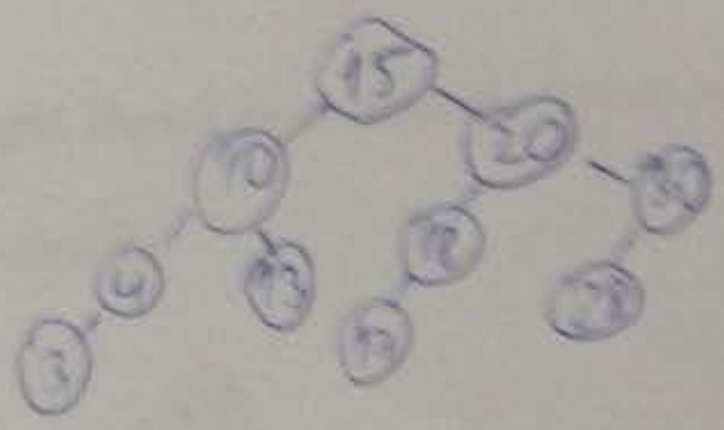
I 2 Uvažujte prázdnu rozptylovú tabuľku, ktorej veľkosť je 6 (miest pamäti) a rozptylová funkcia je h(x) = x mod 6, pričom kolízie sa riešia zreťazením. Nakreslite náčrt stavu po vložení postupnosti prvkov (kľúčov) 35, 2, 18, 6, 3, 10, 9, 5.
(Nekreslite priebežné stavy.)

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.





L 2 Toto je pole práve po prvom rozčlenení v algoritme rýchleho usporadávania (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