Níže uvedené úlohy představují přehled otázek, které se vyskytly v tomto nebo v minulých semestrech ve cvičení nebo v minulých semestrech u zkoušky. Mezi otázkami semestrovými a zkouškovými není žádný rozdíl, předpokládáme, že připravený posluchač dokáže zdárně zodpovědět většinu z nich.

Tento dokument je k dispozici ve variantě převážně s řešením a bez řešení.

Je to pracovní dokument a nebyl soustavně redigován, tým ALG neručí za překlepy a jazykové prohřešky, většina odpovědí a řešení je ale pravděpodobně správně :-).

---------- HASHING GENERAL ----------

1.
Hashovací (=rozptylovací) funkce

a) převádí adresu daného prvku na jemu příslušný klíč
b) vrací pro každý klíč jedinečnou hodnotu
c) pro daný klíč vypočte adresu
d) vrací pro dva stejné klíče různou hodnotu

2.
Kolize u hashovací (rozptylovací) funkce h(k)
a) je situace, kdy pro dva různé klíče k vrátí h(k) stejnou hodnotu
b) je situace, kdy pro dva stejné klíče k vrátí h(k) různou hodnotu
c) je situace, kdy funkce h(k) při výpočtu havaruje
d) je situace, kdy v otevřeném rozptylování dojde dynamická paměť

3.
Hashovací (=rozptylovací) funkce

e) převádí adresu daného prvku na jemu příslušný klíč
f) vrací pro každý klíč jedinečnou hodnotu
g) pro daný klíč vypočte adresu
h) vrací pro dva stejné klíče různou hodnotu

---------- HASHING CHAINED ----------

4.
Implementujte operace Init, Search, Insert a Delete pro rozptylovací tabulku se zřetězeným rozptylováním, do níž se ukládají celočíselné klíče. Předpokládejte, že rozptylovací funkce je již implementována a Vám stačí ji jen volat.

5.
A elsewhere
Hash table of size *m* in hashing with chaining contains *n* elements (keys). Its implementation optimizes the *Insert* operation. The worst case of insertion a new element has the complexity

a) $\Theta(n)$
b) $\Theta(m)$
c) $\Theta(m/n)$
d) $O(1)$
e) $\Theta(\log(n))$

6.
A where?
Linked list of synonyms
a) minimizes the overall cluster length in open address hashing method
b) solves the problem of collisions by inserting the key to the first empty space in the array
c) is a sequence of synonyms stored in continuous segment of addresses
d) does not exist in open address hashing

7.
Zřetězený seznam synonym
e) minimalizuje délku clusterů u metody otevřeného rozptylování
f) řeší kolize uložením klíče na první volné místo v poli
g) je posloupnost synonym uložená v souvislém úseku adres
h) u otevřeného rozptylování nevzniká

8.
Metoda hashování s vnějším zřetězením
a) nemá problém s kolizemi, protože při ní nevznikají
b) dokáže uložit pouze předem známý počet klíčů
c) ukládá synonyma do samostatných seznamů v dynamické paměti
d) ukládá synonyma spolu s ostatními klíči v poli

9.
Metoda hashování s vnějším zřetězením
a) nemá problém s kolizemi, protože nevznikají
b) řeší kolize uložením klíče na první volné místo v poli
c) dokáže uložit pouze předem známý počet klíčů
d) dokáže uložit libovolný předem neznámý počet klíčů

10.
Metoda hashování s vnějším zřetězením
- nemá problém s kolizemi, protože při ní nevznikají
- dokáže uložit pouze předem známý počet klíčů
- ukládá synonyma do samostatných seznamů v dynamické paměti
- ukládá synonyma spolu s ostatními klíči v poli

11.
Rozptylovací tabulka o velikosti *m* se zřetězeným rozptylováním obsahuje *n* prvků. Nejhorší případ, který může při vložení dalšího prvku nastat, má složitost

- o $\Theta(n)$
- o $\Theta(m)$
- o $\Theta(m/n)$
- ✓ $O(1)$
- o $\Theta(\log(n))$

12.
Implementujte operace Init, Search, Insert a Delete pro rozptylovací tabulku se zřetězeným rozptylováním, do níž se ukládají celočíselné klíče. Předpokládejte, že rozptylovací funkce je již implementována a Vám stačí ji jen volat.

---------- HASHING OPEN ----------

13.
Metoda otevřeného rozptylování
a) generuje vzájemně disjunktní řetězce synonym
b) dokáže uložit pouze předem známý počet klíčů
c) zamezuje vytváření dlouhých clusterů ukládáním synonym do samostatných seznamů v dynamické paměti
d) dokáže uložit libovolný předem neznámý počet klíčů

14.

Metoda otevřeného rozptylování
a) dokáže uložit libovolný předem neznámý počet klíčů
b) nemá problém s kolizemi, protože nevznikají
c) ukládá prvky s klíči v dynamické paměti
d) ukládá prvky do pole pevné délky

15.
Metoda otevřeného rozptylování
- generuje vzájemně disjunktní řetězce synonym
- dokáže uložit pouze předem známý počet klíčů
- zamezuje vytváření dlouhých clusterů ukládáním synonym do samostatných seznamů v dynamické paměti
- dokáže uložit libovolný předem neznámý počet klíčů

16.
Rozptylovací tabulka o velikosti *m* s otevřeným rozptylováním obsahuje *n* prvků. Při vložení (n+1)-ého prvku nastala kolize. To znamená, že

a) n = m
b) n > m
c) n = m mod n
d) m = n mod m
e) nic z předchozího

17.
Hash table of size *m* in open address hashing contains *n* elements (keys). While inserting the $(n+1)^{th}$ element a collision appeared. That means:

f) *n* = *m*
g) *n* > *m*
h) *n* = *m* mod *n*
i) *m* = *n* mod *m*
j) none of these answers

18.
A Where?
The hash table uses the hash function (x) = x mod 6 and it was originally empty. Then the following elements were inserted into the table and one collision occured. Which elements?

a) 6 12 24
b) 24 6 12
c) 1 7 6
d) 5 6 7
e) 2 3 4

19.
The hash table uses the hash function (x) = x mod 5 and it was originally empty. Then the following elements were inserted into the table and one collision occured. Which elements?

a) 5 6 7
b) 10 15 20
c) 20 10 15
d) 5 6 11
e) 3 6 9

20.
The word “cluster“ used in open hashing means the following

a) a sequence of synonyms stored in a continuous area of addresses
b) a sequence of keys stored in a continuous area of addresses
c) a sequence of synonyms stored in the dynamic memory
d) nothing, clusters does not appear in the open hashing

21.
In open address hashing

a) unlimited number of synonyms can be stored
b) the range of keys must be defined
c) the array must be extended after a given number of collisions
d) number of stored elements is limited by the array size

22.
Kolize při vkládání klíče do rozptylovací tabulky s otevřeným rozptylováním znamená, že:
- o klíč nebude možno do tabulky vložit
- o klíč bude možno do tabulky vložit po jejím zvětšení
- ✓ místo pro klíč v poli je již obsazeno jiným klíčem
- o v paměti není dostatek místa pro zvětšení tabulky
- o kapacita tabulky je vyčerpána

23.
V otevřeném rozptylování
e) je nutno definovat rozsah hodnot klíčů
f) je počet uložených prvků omezen velikostí pole
g) je nutno po určitém počtu kolizí zvětšit velikost pole
h) je možno uložit libovolný počet synonym

24.
Cluster (u metody otevřeného rozptylování)
a) je posloupnost synonym uložená v souvislém úseku adres
b) je posloupnost klíčů uložená v souvislém úseku adres
c) je posloupnost synonym uložená v dynamické paměti
d) u otevřeného rozptylování nevzniká

25.
Implementujte operace Init, Search, Insert pro rozptylovací tabulku s otevřeným rozptylováním, do níž se ukládají celočíselné klíče. Předpokládejte, že rozptylovací funkce je již implementována a Vám stačí ji jen volat. Použijte strategii „Linear probing“.

---------- HASHING LINEAR ----------

26.
Pole, ve kterém je uložena rozptylovací tabulka vypadá při použití rozptylovací funkce h(k) = k mod 5, lineárního prohledávání (linear probing) a vložení klíčů
8, 9, 4, 3 (vkládaných v pořadí zleva doprava) takto

0	1	2	3	4
4	3		8	9

a)

0	1	2	3	4
8	9	4	3	

b)

0	1	2	3	4
8	9		3	4

c)

0	1	2	3	4
	9	8	3	4

d)

27.
Pole, ve kterém je uložena rozptylovací tabulka vypadá při použití rozptylovací funkce h(k) = k mod 5, lineárního prohledávání (linear probing) a vložení klíčů 7, 1, 6, 2 (vkládaných v pořadí zleva doprava) takto

0	1	2	3	4
7	1	6	2	

a)

0	1	2	3	4
6		7	1	2

b)

0	1	2	3	4
	1	7	6	2

c)

0	1	2	3	4
	6	2	1	7

d)

28.
A hash table is stored in an array. The keys inserted into the originally empty table are 7, 1, 6, 2. The table uses hash function h(k) = k mod 5 and resolves collisïons by linear probing scheme. What is the resulting contents of the table?

0	1	2	3	4
7	1	6	2	

a)

0	1	2	3	4
6		7	1	2

b)

0	1	2	3	4
	1	7	6	2

c)

0	1	2	3	4
	6	2	1	7

d)

29.
Pole, ve kterém je uložena rozptylovací tabulka vypadá při použití rozptylovací funkce h(k) = k mod 5, lineárního prohledávání (linear probing) a vložení klíčů
5, 9, 4, 6 (vkládaných v pořadí zleva doprava) takto

0	1	2	3	4
5	6	4		9

a)

0	1	2	3	4
5	6	9		4

b)

0	1	2	3	4
5	4	6		9

c)

0	1	2	3	4
4	5	6		9

d)

30. A
Hashing uses linear probing and a hash function $h(k) = k \bmod 5$. We insert the keys
5, 9, 4, 6 (in this order). The array used for storage of the hash table looks then as follows:

0	1	2	3	4
5	6	4		9

a)

0	1	2	3	4
5	6	9		4

b)

0	1	2	3	4
5	4	6		9

c)

0	1	2	3	4
4	5	6		9

d)

31.
Pole, ve kterém je uložena rozptylovací tabulka vypadá při použití rozptylovací funkce h(k) = k mod 5, lineárního prohledávání (linear probing) a vložení klíčů
4, 5, 9, 6 (vkládaných v pořadí zleva doprava) takto

0	1	2	3	4
5	6	4		9

a)

0	1	2	3	4
5	9	6		4

b)

0	1	2	3	4
4	6	5		9

c)

0	1	2	3	4
4	5	6		9

d)

32. A
Hashing uses linear probing and a hash function $h(k) = k \bmod 5$. We insert the keys
4, 5, 9, 6 (in this order). The array used for storage of the hash table looks then as follows:

0	1	2	3	4
5	6	4		9

a)

0	1	2	3	4
5	9	6		4

b)

0	1	2	3	4
4	6	5		9

c)

0	1	2	3	4
4	5	6		9

d)

33.
Pole, ve kterém je uložena rozptylovací tabulka vypadá při použití rozptylovací funkce h(k) = k mod 5, lineárního prohledávání (linear probing) a vložení klíčů 6, 5, 9, 4 (vkládaných v pořadí zleva doprava) takto

a)

0	1	2	3	4
5	6	4		9

b)

0	1	2	3	4
5	6	9		4

c)

0	1	2	3	4
4	6	5		9

d)

0	1	2	3	4
4	5	6		9

34.
Pole, ve kterém je uložena rozptylovací tabulka vypadá při použití rozptylovací funkce h(k) = k mod 5, lineárního prohledávání (linear probing) a vložení klíčů 6, 4, 5, 9 (vkládaných v pořadí zleva doprava) takto

a)

0	1	2	3	4
5	6	4		9

b)

0	1	2	3	4
5	6	9		4

c)

0	1	2	3	4
4	6	5		9

d)

0	1	2	3	4
4	5	6		9

35.
Implementujte operace Init, Search, Insert pro rozptylovací tabulku s otevřeným rozptylováním, do níž se ukládají celočíselné klíče. Předpokládejte, že rozptylovací funkce je již implementována a Vám stačí ji jen volat. Použijte strategii „Linear probing“.

---------- HASHING DOUBLE ----------

36.
Double hashing

a) je metoda ukládání klíčů na dvě různá místa současně
b) je metoda minimalizace kolizí u metody otevřeného rozptylování
c) má vyšší pravděpodobnost vzniku kolizí než linear probing
d) je metoda minimalizace kolizí u metody rozptylování s vnějším zřetězením

37.
Double hashing

a) má stejnou pravděpodobnost vzniku dlouhých clusterů jako linear probing
b) je metoda ukládání klíčů na dvě různá místa
c) je metoda minimalizace délky clusterů u metody otevřeného rozptylování
d) má vyšší pravděpodobnost vzniku dlouhých clusterů než linear probing

---------- HASHING COALESCED ----------

N

38.
Uložte dané klíče v daném pořadí postupně do rozptylovací tabulky. Porovnejte počet kolizí při ukládání klíčů do tabulek různé velikosti a použití různých strategií pro srůstání řetězců kolidujících klíčů: LISCH, LICH, EISCH, EICH.

```
LISCH - Late Insert Standard Coalesced Hashing
Keys to insert:  9 11 18 27 29 36 43 45
Table size: 9
Hash function: h(k) = k % 9
```

```
LISCH - Late Insert Standard Coalesced Hashing
Keys to insert:  10 12 20 23 32 39 40
Table size: 10
Hash function: h(k) = k % 10
```

39.
Oba předchozí případy zopakujeme pro stejná data, pouze použijeme tabulku se „sklepem“ o velikosti 2, přičemž celková velikost tabulky se nezmění.

```
LICH - Late Insert Coalesced Hashing
Keys to insert:  9 11 18 27 29 36 43 45
Table size: 7    Cellar size: 2
Hash function: h(k) = k % 7

LICH - Late Insert Coalesced Hashing
Keys to insert:  10 12 20 23 32 39 40
Table size: 8    Cellar size: 2
Hash function: h(k) = k % 8
```

40.
Oba předchozí případy zopakujeme pro stejná data, použijme metodu EISCH , přičemž celková velikost tabulky se nezmění.

```
EISCH - Early Insert Standard Coalesced Hashing
Keys to insert:  9 11 18 27 29 36 43 45
Table size: 9
Hash function: h(k) = k % 9


EISCH - Early Insert Standard Coalesced Hashing
Keys to insert:  10 12 20 23 32 39 40
Table size: 10
Hash function: h(k) = k % 10
```

Oba předchozí případy nakonec zopakujeme pro stejná data, použijme metodu EICH a tabulku se „sklepem“ o velikosti 2, přičemž celková velikost tabulky se nezmění.

```
EICH - Early Insert Coalesced Hashing
Keys to insert:  9 11 18 27 29 36 43 45
Table size: 7    Cellar size: 2
Hash function: h(k) = k % 7

EICH - Early Insert Coalesced Hashing
Keys to insert:  10 12 20 23 32 39 40
Table size: 8    Cellar size: 2
Hash function: h(k) = k % 8
```

41.
Pro data
 9 11 18 27 29 36 43 45
jsme použitím metod LISCH, LICH, EISCH, EICH získali čtyři různé tabulky stejné velikosti, které pro přehled opakujeme níže. Předpokládejme, že v tabulce budeme vzhledávat vždy pouze klíče, které tam jsou uloženy, přičemž frekvence hledání budou pro všechny klíče stejné (= všechny klíče budeme vyhledávat stejně často). Která z uvedených tabulek je z tohoto hlediska nejvýhodnější?

```
LISCH - Late Insert Standard Coalesced Hashing
  0  1  2  3  4  5  6  7  8
  9  - 11 45 43 36 29 27 18
  8  .  6  .  3  4  .  5  7
```

```
LICH - Late Insert Coalesced Hashing
  0  1  2  3  4  5  6 |  7  8
  - 29  9 45 11 43 27 | 36 18
  .  7  .  .  8  .  . |  5  .

EISCH - Early Insert Standard Coalesced Hashing
  0  1  2  3  4  5  6  7  8
  9  - 11 45 43 36 29 27 18
  3  .  6  5  8  7  .  4  .

EICH - Early Insert Coalesced Hashing
  0  1  2  3  4  5  6 |  7  8
  - 29  9 45 11 43 27 | 36 18
  .  5  .  .  8  7  . |  .  .
```

42.
Předchozí úlohu zopakujeme pro data
10 12 20 23 32 39 40
a jim příslušné čtyři tabulky o velikosti 10 a případné velikosti “sklepa“ 2.