BBM204: Software Practicum Quiz #2

Hacettepe University Computer Engineering Department

April 6, 2021

1. Select the correct hash function(s) to hash to a value (key.hashCode) between 0 and M-1.

   A. key.hashCode() % M

   B. Math.abs(key.hashCode()) % M

   C. (key.hashCode() & 0x7fffffff) % M

   D. (key.hashCode() % M) & 0x7fffffff

> **Solution:** key.hashCode() method can return a negative integer. For this reason choice A cannot be selected. Math.abs() can also return negative when the argument is Integer.MIN_VALUE, therefore choice B cannot be selected neither. In choice D, the piece of code will not return a value within the wanted range for negative numbers (you can try it yourself using the following line
>
> ```
> System.out.println((-3 % 41) & 0x7FFFFFFF);
> ```
>
>). In choice C, there is a bit-mask corresponding to the Integer.MAX_VALUE, this mask makes sure the number is not negative and the modulus operation that follows makes sure the number is in the wanted range. Please read [1] and [2].

2. **T** Insertion in hashing with the linear probing has O(N) worst-case complexity.

> **Solution:** Considering the worst case, the program will have to iterate over all array elements resulting in O(N) complexity.

3. **F** With the separate chaining approach, the worst-case cost of deletion in a hash table is O(1).

Solution: Considering the worst case, all the keys will get the same hash value therefore accumulating on a single chain. The cost of finding an element in that long chain will be O(N).

4. Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming simple uniform hashing, what is the expected number of colliding pairs of elements is in h? Hint: Simple uniform hashing means that the probability of element i hashing to slot k is $1/m$.

   A. $O(n)$
   B. $O(n/m)$
   C. $O(n^2)$
   D. $O(n^2/m)$

Solution:

$$\begin{aligned}
\text{First insertion:} \quad & P_{collision} = 0 \\
\text{Second insertion:} \quad & P_{collision} = 1/m \\
\text{Third insertion:} \quad & P_{collision} = 2/m \\
\text{Fourth insertion:} \quad & P_{collision} = 3/m \\
& \dots \\
N^{th} \text{ insertion:} \quad & P_{collision} = (n-1)/m
\end{aligned}$$

$$\begin{aligned}
\text{Total Probability} &= \frac{n(n-1)}{2m} \\
&= \frac{n^2 - n}{2m} \\
&= O(n^2/m)
\end{aligned}$$

Please also read [3] and [4]

5. **F** "Dynamic Programming" can be used interchangeably with "Divide and Conquer Approach".

Solution: Every divide and conquer problem is not a dynamic programming problem while every dynamic programming problem is a divide and conquer problem. Therefore the given statement is false. Please also read [5]

6. **F** Merge-sort algorithm can be optimized using dynamic programming.

Solution: It does not have overlapping subproblems. Therefore the given statement is false.

7. The following keys are inserted into an initially empty linear-probing hash table.

key	hash
D	5
F	0
K	5
N	1
P	3
R	5
X	4

Suppose that the size of the hash table is fixed as 7 and the insertions are not necessarily in the order given above. Please specify which one or more of the following could be the contents of the resulting array? You must mark all required answers to get full credits.

A. R N D F P K X

B. X K N D R P F

C. F N R P X K D

D. X N K F P D R

E. D F N P X R K

Solution: Considering the probable collisions in the given elements, only the elements with the keys D, K and R will collide. It is only possible to encounter those 3 elements in the indices 5 and 6. Therefore the choices A and B are incorrect. There is no reason for X to be at the first index. This is because the collisions that may occur before X, cannot go far enough to cause X's position to be shifted. Therefore answer D is also incorrect. The same thing can be said about the element P.

For the choices C and E, possible insertion sequences can be seen below:

For the choice C:

insert K → index 5
insert F → index 0
insert D → index 6
insert N → index 1
insert P → index 3
insert R → index 2
insert X → index 4

```
F N R P X K D
0 1 2 3 4 5 6
```

For the choice E:

insert R → index 5
insert K → index 6
insert D → index 0
insert F → index 1
insert N → index 2
insert P → index 3
insert X → index 4

```
D F N P X R K
0 1 2 3 4 5 6
```

8. Suppose that the following keys are inserted in the given order into a separate-chaining hash table with 3 chains.

key	hash
I	0
B	1
G	2
C	1
V	2
O	0
X	1
M	0
F	1
W	1
Y	2
L	0

Consider one searches for the key C. Please specify the sequence of keys that are compared during this search. Your answer should be a sequence of letters separated by whitespace, e.g. I B G.

Solution: As specified in the course slides [6] (slide 17), elements are inserted at front of the i^{th} chain for element X, where $H(X) = i$.

So the chain looks like:
0 L → M → O → I
1 W → F → X → C → B
2 Y → V → G

Therefore the solution will be: W F X C

```
1   public class Search{
2
3       public static void main(String[] args) {
4           int CONTEXT = 5;
5
6           In in = new In(args[0]);
7           String[] words = in.readAllStrings(); // reads all words
8           (A)
9
10          // build up concordance
11          for (int i = 0; i < words.length; i++) {
12              String s = words[i];
13              if (!st.contains(s)) {
14                  (B)
15              }
16              SET<Integer> set = st.get(s);
17              set.add(i);
18          }
19          StdOut.println("Finished building concordance");
20
21          // process queries
22          while (!StdIn.isEmpty()) {
23              String query = StdIn.readString();
24              (C)
25              if (set == null) set = new SET<Integer>();
26              for (int k : set) {
27                  for (int i = Math.max(0, k - CONTEXT + 1); i < k; i++)
28                      StdOut.print(words[i] + " ");
29                  StdOut.print("*" + words[k] + "* ");
30                  for (int i = k + 1; i < Math.min(k + CONTEXT, words.length); i++)
31                      StdOut.print(words[i] + " ");
32                  StdOut.println();
33              }
34              StdOut.println();
35          }
36
37      }
38  }
39  }
```

9. Given the above implementation of the Search class using the symbol table ST ADT, complete the missing statements (A), (B), and (C).

   (a) (A)

   Solution:

   ```
   ST<String, SET<Integer>> st = new ST<String, SET<Integer>>();
   ```

As the variable *st* is later used in the code, it should be instantiated before. This variable will store an object of the symbol table to be filled so it can be used for querying later. As *ST* is a generic class, it should be defined and instantiated using types, those types can be inferred using lines 13 and 16.

(b) (B)

Solution:

```
st.put(s, new SET<Integer>());
```

If the symbol table does not contain the string *s*, an integer set should be instantiated for this string. This set will contain elements which are put in line 17.

(c) (C)

Solution:

```
SET<Integer> set = st.get(query);
```

It can be inferred from the line 25, a variable named *set* needs to be defined and as the null check indicates, it should contain some data. The usage of the *get* method is already have been demonstrated in line 16. Please also read the course slides [6] (slides 106-118) if you have not done already.

10. Considering gap penalty = 1, mismatch penalty = 1, and match penalty = 0, please estimate the cost of the optimal alignment between the strings PARK and SPAKE.

Solution:

		s	p	a	k	e
	0	**1**	2	3	4	5
p	1	1	**1**	2	3	4
a	2	2	2	**1**	2	3
r	3	3	3	2	**2**	3
k	4	4	4	3	2	**3**

As can be seen from the above table, cost of the optimal alignment is **3** with the following possible alignments: (An interesting tool about this topic can be found at [7])

```
- p a r k
s p a k e

- p a r k -
s p a - k e
```

11. Given a knapsack with a maximum capacity of 10, and 5 distinct items with the following characteristics:

item	value	weight
1	20	6
2	8	2
3	14	4
4	13	3
5	35	11

Answer the following questions by considering the partial solutions suggested by the bottom-up dynamic programming algorithm covered in the class.

(a) What will be the optimal value of knapsack problem with items 1, . . . , 3 subject to the weight limit 7.

> **Solution:** When the items 2 and 3 are included, the maximum possible value is achieved, which is **22**.

(b) What will be the optimal value of knapsack problem with items 1, . . . , 4 subject to the weight limit 9.

> **Solution:** When the items 2, 3 and 4 are included, the maximum possible value is achieved, which is **35**.

References

[1] https://www.tutorialspoint.com/java/lang/math_abs_int.htm

[2] https://stackoverflow.com/questions/9249983/hashcode-giving-negative-values

[3] https://gateoverflow.in/57653/cormen-2nd-edition-exercise-11-2-1

[4] https://www.slader.com/discussion/question/suppose-we-use-a-hash-function-h-to-hash-n-distinct-keys-into-an-array-t-of-length-m-assuming-simple/

[5] https://stackoverflow.com/questions/13538459/difference-between-divide-and-conquer-algo-and-dynamic-programming

[6] https://web.cs.hacettepe.edu.tr/~bbm202/slides/07-hashing-apps.pdf

[7] http://www.let.rug.nl/~kleiweg/lev/