Algorithms & Data Structures, Class № 11

Hash tables

Many applications require a dynamic set that supports only the dictionary operations INSERT, SEARCH, and DELETE. Example: a symbol table in a compiler.

A hash table is effective for implementing a dictionary.

- The expected time to search for an element in a hash table is $O(1)$, under some reasonable assumptions.
- Worst-case search time is $\Theta(n)$, however.

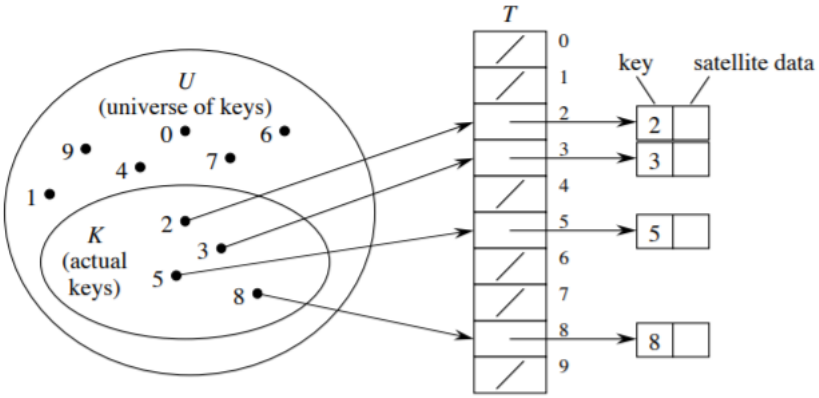
A hash table is a generalization of an ordinary array.

- With an ordinary array, we store the element whose key is k in position k of the array.
- Given a key k, we find the element whose key is k by just looking in the k-th position of the array. This is called direct addressing.
- Direct addressing is applicable when we can afford to allocate an array with one position for every possible key.

We use a hash table when we do not want to (or cannot) allocate an array with one position per possible key.

- Use a hash table when the number of keys actually stored is small relative to the number of possible keys.
- A hash table is an array, but it typically uses a size proportional to the number of keys to be stored (rather than the number of possible keys).
- Given a key k, don't just use k as the index into the array. Instead, compute a *function of* k, and use that value to index into the array. We call this function a hash function.

Direct Access Table



Dictionary operations are trivial and take O(1) time each.

Hash tables

Instead of storing element with key k in slot k, use a function h and store the el. in slot $h(k)$.

$$h : U \to \{0, 1, \ldots, m-1\}$$

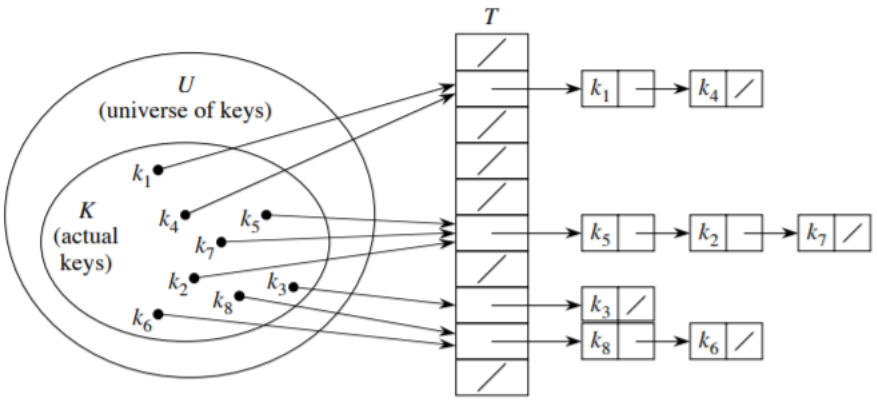
so that $h(k)$ is a legal slot number in T.

Exemplary hashing functions:

- Division method: $h(k) = k \bmod m$
- Multiplication method: $h(k) = m(kA \bmod 1)$, where $kA \bmod 1 = kA - \lfloor kA \rfloor$ = fractional part of kA.

Collisions: When two or more keys hash to the same slot – can happen when there are more possible keys than slots ($|U| > m$).

Collision resolution by chaining




```
1  CHAINED-HASH-INSERT(T, x)
2    insert x at the head of list T [h(key[x])]
```

Worst-case running time is O(1). Assumes that the element being inserted isn't already in the list. It would take an additional search to check if it was already inserted.

```
1  CHAINED-HASH-SEARCH(T, k)
2    search for an element with key k in list T [h(k)]
```

Running time is proportional to the length of the list of elements in slot h(k).

```
1  CHAINED-HASH-DELETE(T, x)
2    delete x from the list T [h(key[x])]
```

Collision resolution by probing (open addressing)

- linear probing: $h(k,i) = (h'(k) + i) \, mod \, m$
- quadratic probing: $h(k,i) = (h'(k) + c_1 i + c_2 i^2) \, mod \, m$
- double hashing: $h(k,i) = (h_1(k) + i \, h_2(k)) \, mod \, m$.

```
1  HASH-SEARCH(T, k)
2    i := 0
3    repeat
4      j := h(k,i)
5      if T [j] = k
6        then return j
7      i := i + 1
8    until T [j] = NIL or i = m
9  return NIL
```

HASH-SEARCH returns the index of a slot containing key k, or NIL if the search is unsuccessful.

```
1  HASH-INSERT(T, k)
2    i := 0
3    repeat
4      j := h(k,i)
5      if T [j] = NIL
6        then T [j] := k
7        return j
8      else
9        i := i + 1
10   until i = m
11   error "hash table overflow"
```

Problem 1

Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function $x \, mod \, 10$, which of the following statements are true?

1. 9679, 1989, 4199 hash to the same value
2. 1471, 6171 hash to the same value
3. All elements hash to the same value
4. Each element hashes to a different value

Problem 2

The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function $h(k) = k \, mod \, 10$ and linear probing. What is the resultant hash table?

Problem 3

A hash table of length 10 uses open addressing with hash function $h(k) = k \, mod \, 10$, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below. 4 Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

1. 46, 42, 34, 52, 23, 33
2. 34, 42, 23, 52, 33, 46
3. 46, 34, 42, 23, 52, 33
4. 42, 46, 33, 23, 34, 52

Problem 4

Given a hash table with $m = 11$ entries and the following hash function $h1$ and step function $h2$:

- $h_1(key) = key \bmod m$
- $h_2(key) = key \bmod (m-1) + 1$

Insert the keys 22, 1, 13, 11, 24, 33, 18, 42, 31 in the given order (from left to right) to the hash table using each of the following hash methods:

1. Chaining with h_1: $h(k) = h_1(k)$
2. Linear-Probing with h_1: $h(k,i) = (h_1(k) + i) \bmod m$
3. Double-Hashing with h_1 as the hash function and h_2 as the step function: $h(k,i) = (h_1(k) + i\, h_2(k)) \bmod m$

	Chaining	Linear Probing	Double hashing
0	22 ← 11 ← 33	22	22
1	1		1
2	13 ← 24	13	13
3		11	
4		24	11
5		33	
6			
7	18	18	24
8			
9	42 ← 31	42	
10		31	

Problem 5

For the previous h_1 and h_2, is it OK to use h_2 as a hash function and h_1 as a step function?

Problem 6

Why is it important that the result of the step function and the table size will not have a common divisor?

Problem 7

How to delete an element from a hash table with open addressing? Write pseudocode for HASH-DELETE(T, k) procedure and correct the procedures HASH-SEARCH and HASH-INSERT to take into account the behaviour of the HASH-DELETE procedure.