More on Hashing

CSCI2100 Tutorial 8

Shangqi Lu

Adapted from the slides of the previous offerings of the course

Review on Hash Table

- Given a set of integers S in $[1, U]$
- Main idea: divide S into a number m of disjoint subsets
- Guaranteed
  - Space consumption: $O(n)$
  - Query cost: $O(1)$ in expectation
  - Preprocessing cost: $O(n)$

Review on Hash Table

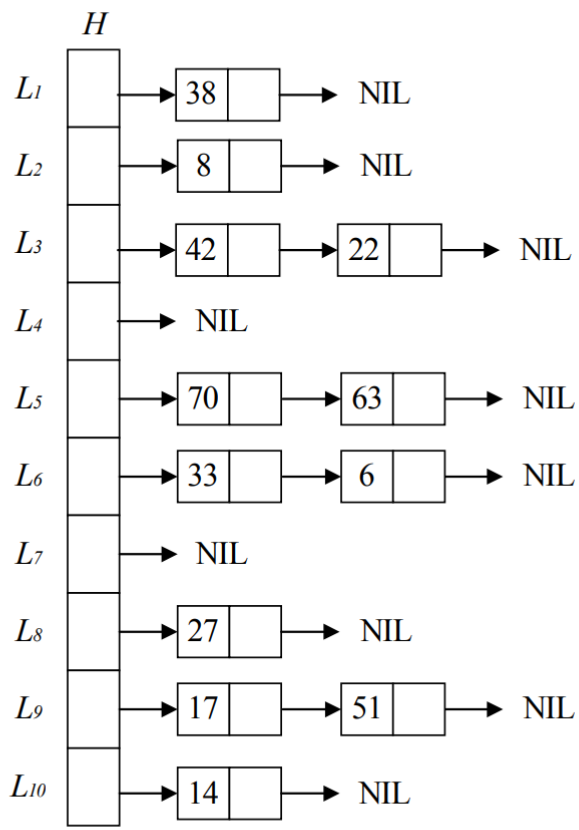
- No single hash function works for all sets
- Construct a hash function from a universal family
  - Pick a prime number p such that $p \geq m$ and $p \geq U$
  - Choose an integer α from $[1, p-1]$ uniformly at random
  - Choose an integer β from $[0, p-1]$ uniformly at random
  - Define a hash function:

$$h(k) = 1 + \big((\alpha k + \beta) \bmod p\big) \bmod m$$

Example

- Let $S = \{33,42,70,38,6,22,17,51,8,14,63,27\}$
- We choose $m = 10, p = 71$, suppose that α and β are randomly chosen to be 3 and 7, respectively
- $h(k) = 1 + \left(\left((3k + 7) \bmod 71\right) \bmod 10\right)$

k	$h(k)$
33	6
42	3
70	5
38	1
6	6
22	3
17	9
51	9
8	2
14	10
63	5
27	8



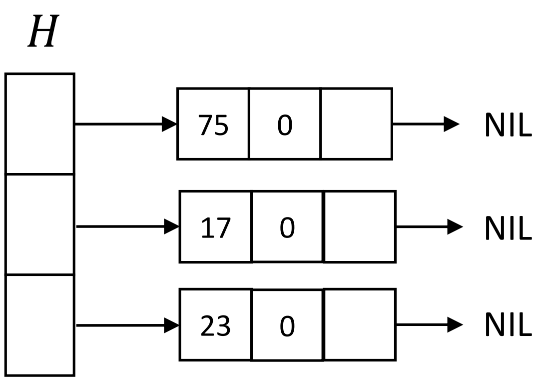
Adapted from Dan He's slides

Regular Exercise 7 Problem 3

- Let S be a multi-set of n integers
- Frequency of an integer x:
  - No. of occurrences of x in S
- Design an algorithm to produce an array that sorts the distinct integers by frequency in $O(n)$ expected time
- E.g.,
  - $S = \{23,75,17,17,23\}$
  - You should output $(75,17,23)$ or $(75,23,17)$
  - If two integers have the same frequency, their relative order is not important

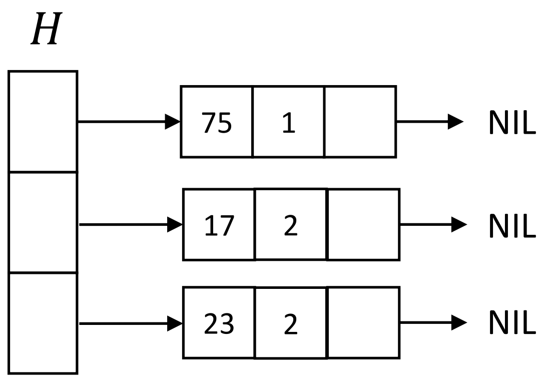
Solution

- First, choose a hash function h and create a hash table H
- For each integer $x \in S$
  - If the H already contained a copy of x
    - Ignore x
  - Else
    - Compute $h(x)$
    - Insert $(x, 0)$ into the $H[h(x)]$
- The checking in each iteration takes $O(1)$ in expectation
- Overall: $O(n)$ in expectation



Solution

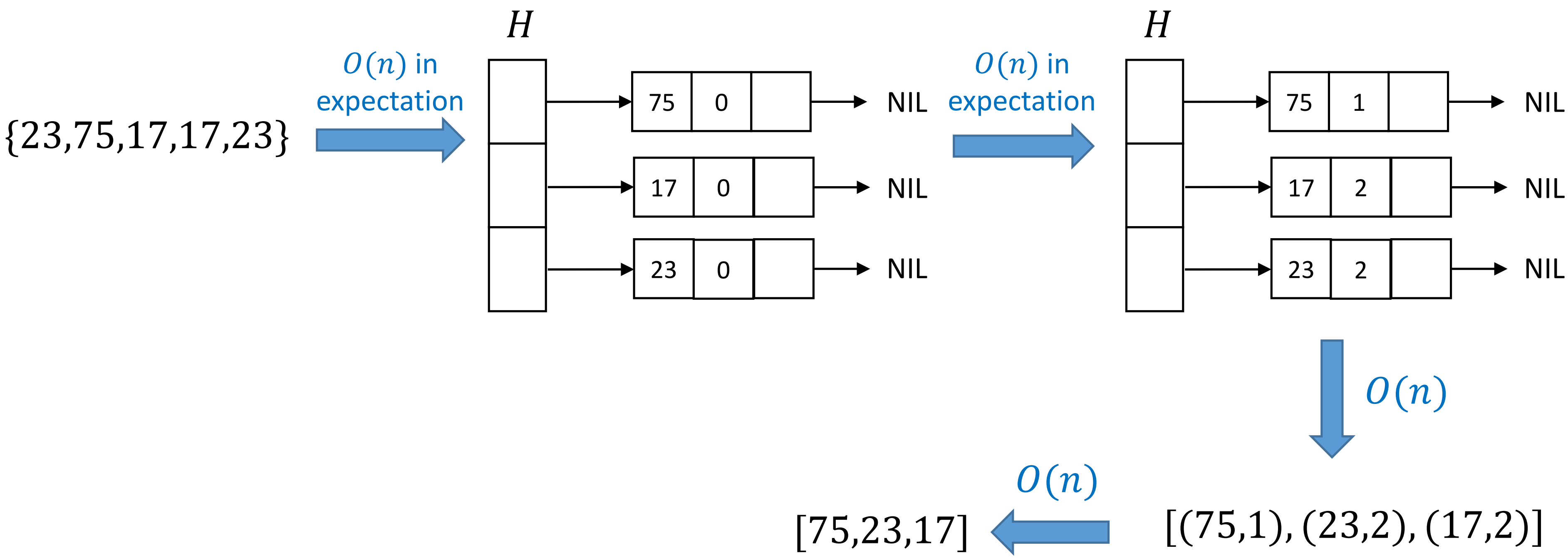
- Second, obtain the frequency of every distinct integers
- For each integer $x \in S$
  - Find its copy in H
  - Increase the counter of the copy by one
  - Takes $O(1)$ expected time
- This part takes $O(n)$ in expectation



Solution

- Finally, sort all the distinct integers by frequency
- Since the frequency of every integer in S is in the domain $[1, n]$
- Use counting sort to sort the integers by frequency (see tutorial 6), takes $O(n)$ time
- E.g., we get $[(75,1), (23,2), (17,2)]$
- Generate output from these sorted tuples, takes $O(n)$ time
- E.g., $[75,23,17]$

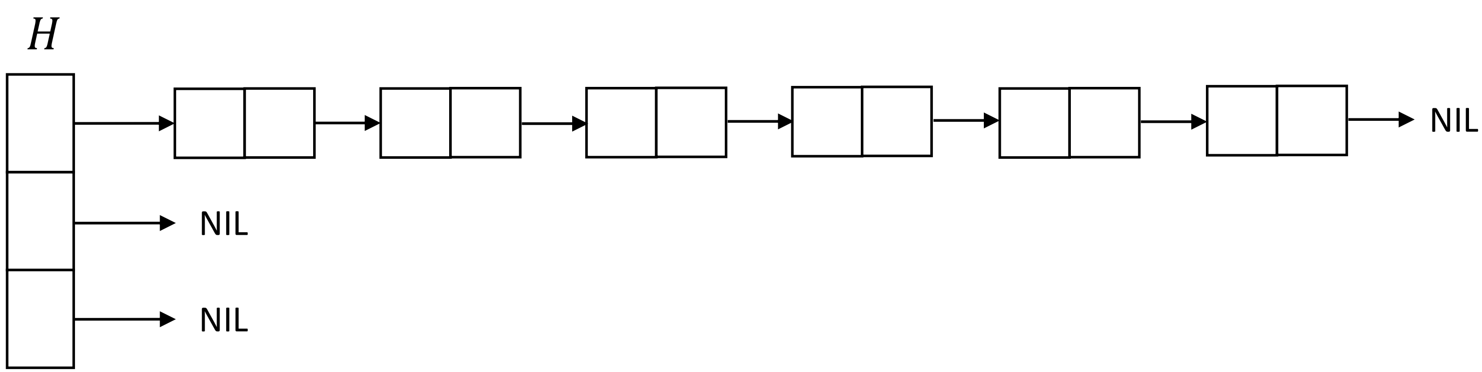
Time Complexity



- Overall complexity: $O(n)$ in expectation

Hash Table

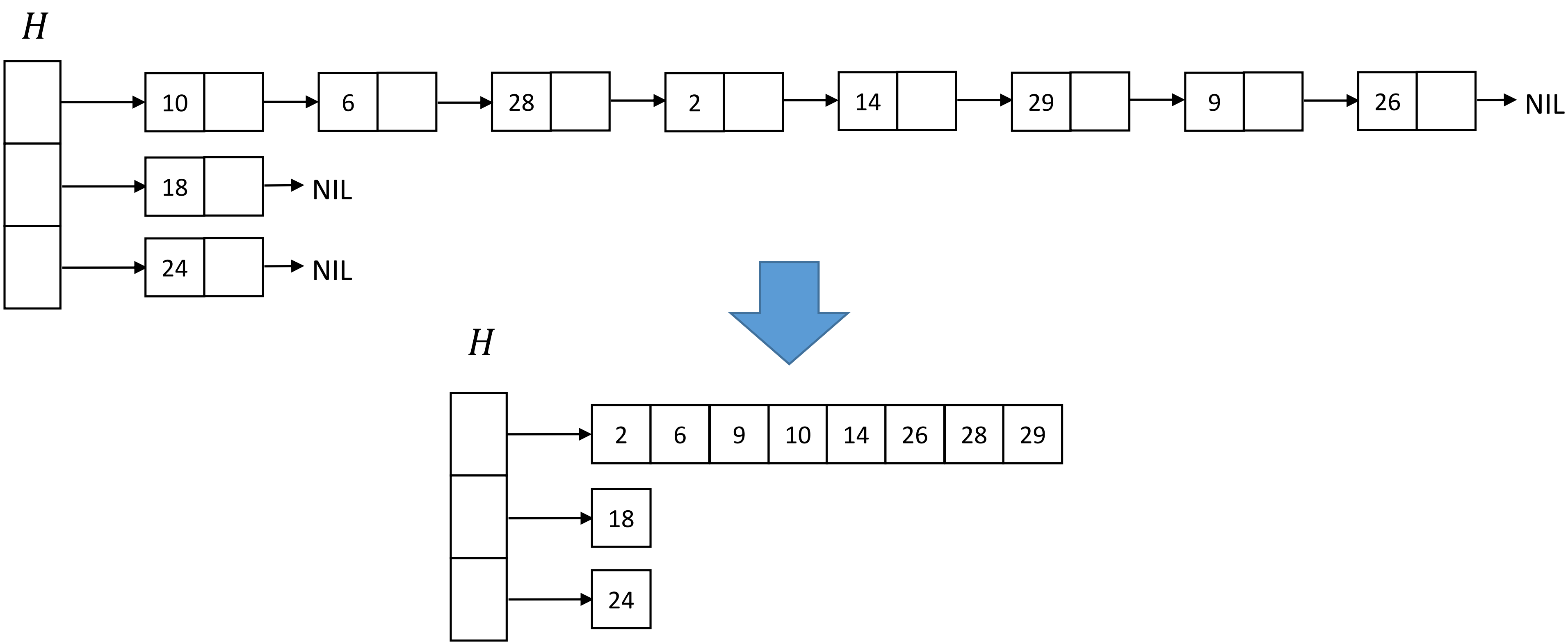
- Expected query cost: $O(1)$
  - Pick a hash function from a universal family
- Worst-case query cost: $O(n)$
  - All elements are hashed into the same value



- Can we improve the worst-case query cost?

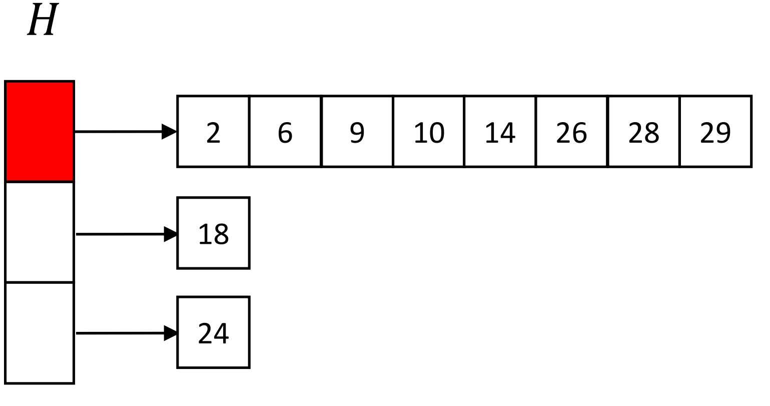
Hash Table

- Replace linked lists with arrays
- Sort the arrays, cost $O(\mathrm{n}\ \mathrm{logn})$ for preprocessing



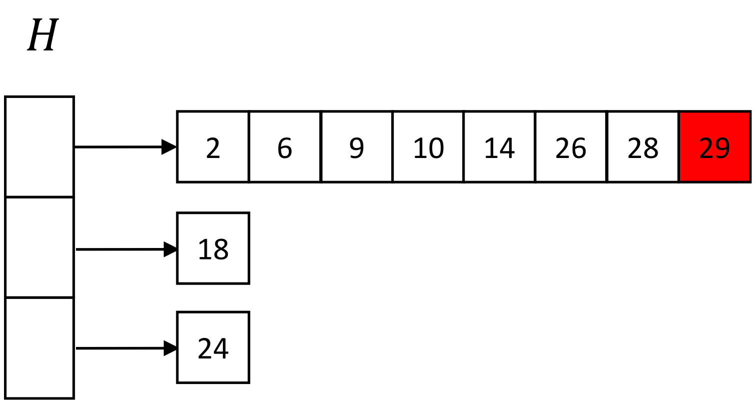
Hash Table

- Query: whether 29 exists
- Step 1:
  - Access the hash table to obtain the address of corresponding array
  - Takes $O(1)$ time



Hash Table

- Query: whether 29 exists
- Step 2:
  - Perform binary search on the array to find the target
  - Takes $O(\log n)$ time
- Overall worst-case complexity: $O(\log n)$



Hash Table

- This method retains the $O(1)$ expected query time
- Proof:
  - Suppose we look up an integer q
  - Define random variable $L_{h(q)}$ to be the length of array that corresponds to the hash value $h(q)$
  - Expected query time:
    - $\mathrm{E}\left[\log_2 L_{h(q)}\right] = \sum_{l=1}^{n} \log_2 l \Pr\left(L_{h(q)} = l\right)$
    - $\leq \sum_{l=1}^{n} l \Pr\left(L_{h(q)} = l\right)$
    - $= \mathrm{E}\left[L_{h(q)}\right]$
    - $= O(1)$

Revisiting the Two-Sum Problem

- Problem Input:
  - A set S of unsorted n distinct integers
  - The value n has been placed in Register 1
  - A positive integer v has been placed in Register 2
- Goal:
  - Determine whether if there exist two different integers x and y in S such that $x + y = v$
- For example:
  - Find a pair whose sum is 20

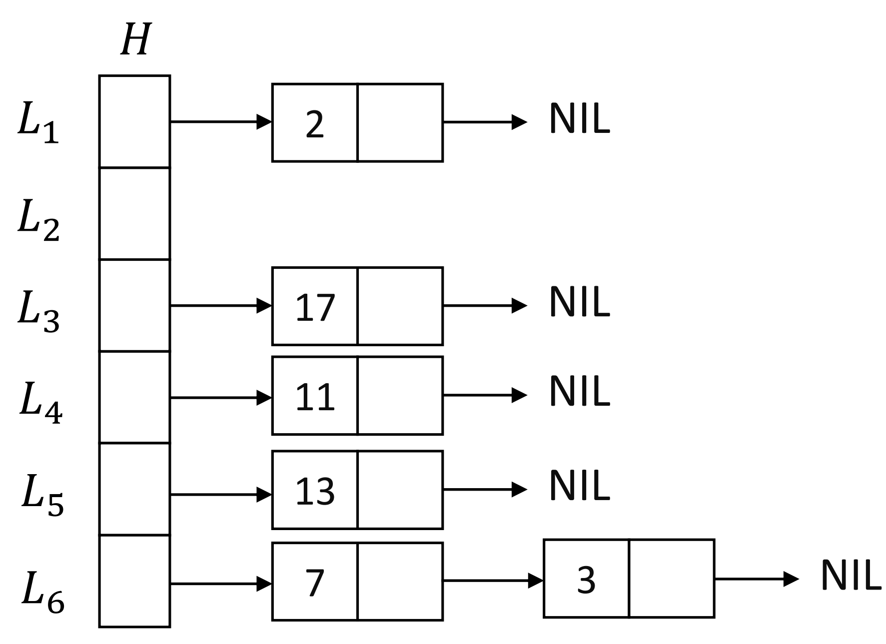
11	3	17	7	2	13

Solution 1: Binary Search the Answer

- Goal: Find $a\ pair(x, y)$ such that $x + y = v$
- Observe that given x, $y = v - x$, is determined
- Solution:
  - Sort S
  - For each x in S:
    - set y as $v - x$
    - Use binary search to see if y exists in the sequence
- Time complexity: $O(n \log_2 n)$

Solution 2: Using the Hash Table

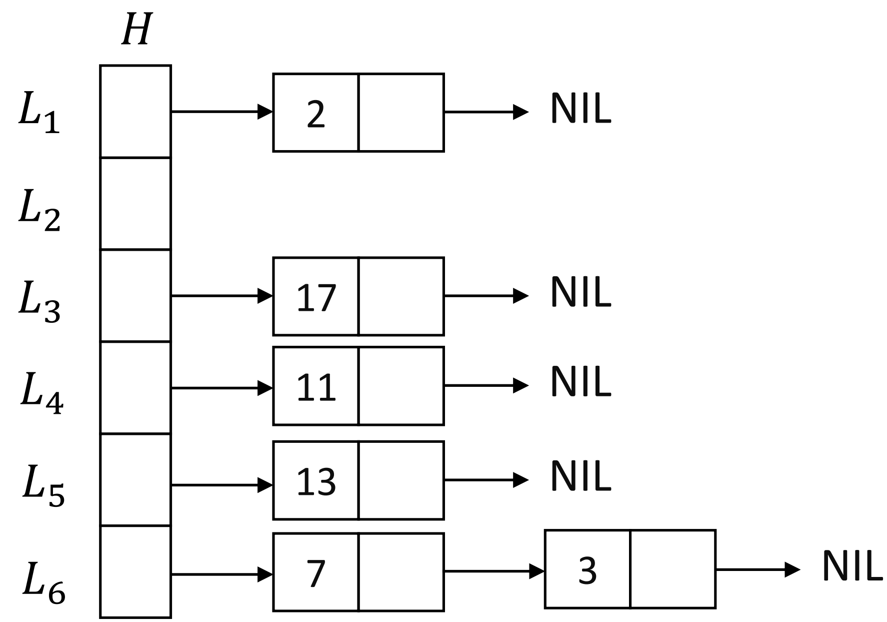
- Step 1 and 2:
  - Choose a hash function h and create an empty hash table H
  - Insert each x in S into $L_{h(x)}$



Solution 2: Using the Hash Table

- Step 3:
  - For each x in S:
    - Set y as $v - x$
    - Check if y is in the hash table
      - If so, return yes
  - Return no

11	3	17	7	2	13



Time Complexity

- Step 1 and 2: $O(n)$
- Step 3: $O(n)$ in expectation
- Overall: $O(n)$ in expectation