COMP20003 Workshop Week 6

Plan

- Hash Tables
- Q 6.1

- Assignment 1: Implementation expectation
- Assignment 1: Report expectation

Hashing

- Concept: hash tables, hash functions
- Comparing: search in array, sorted array, BST, and in hash tables

Collisions

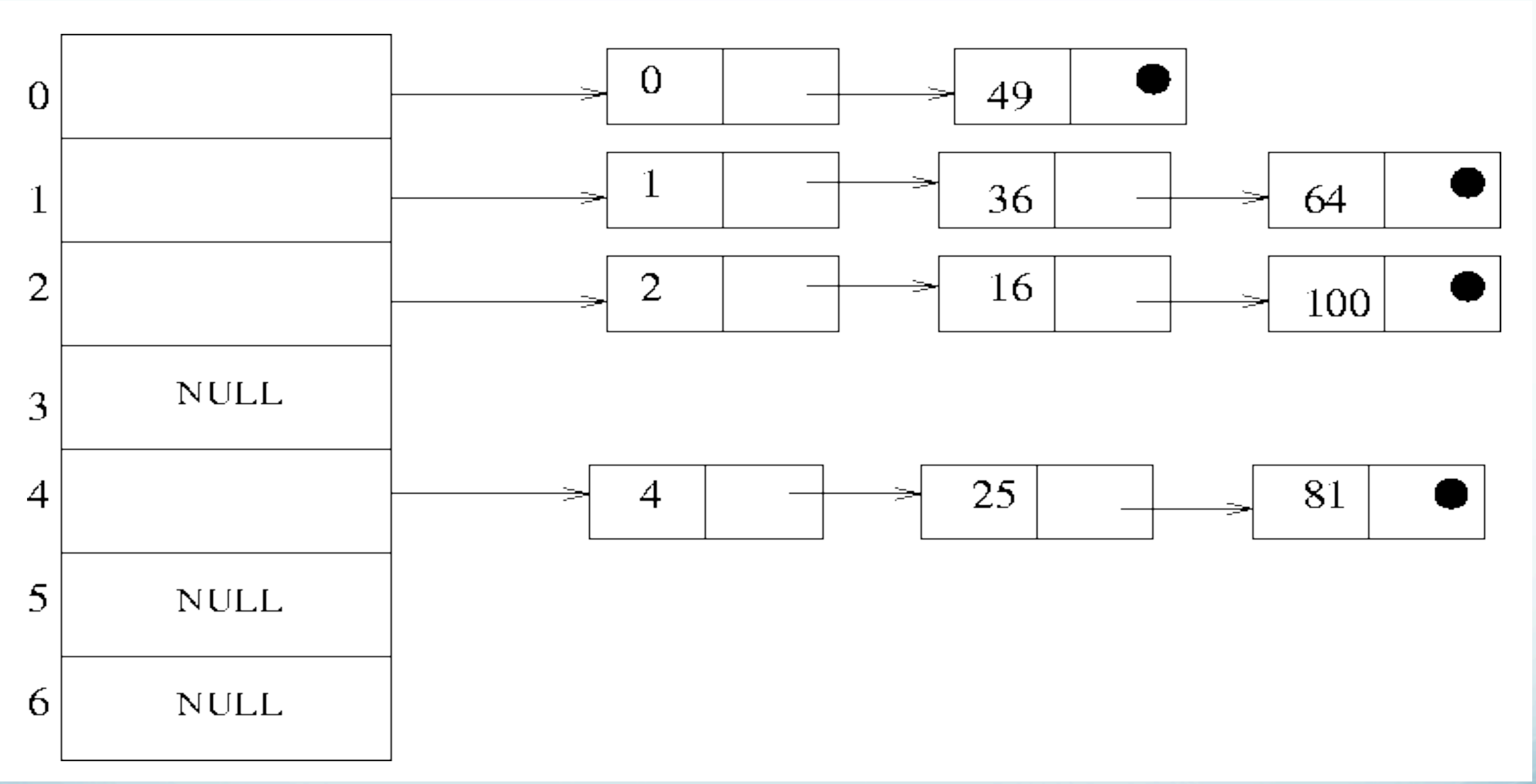
- If managed well, hash tables allow search in *O(1)* time. But collisions are normally unavoidable. Collisions are when two or more keys hashed into a same cell in a hash table, that is, `h(x1) = h(x2)` for some $x1 \neq x2$. (supposing that `h(x)` is the hash function, `x` is a key's value)

- Dealing with collisions is the single biggest problem for hashing.

- One method *to reduce collisions* using a prime number for hash table size.

- Notes: If all key values are known beforehand, we can build a perfect hash function for these keys [not a topic for this subject]

Collision Solution 1: Chaining



Solution 2: Linear Probing

```
while (HT[index] != NULL)
    index= (index+1)%TABLESIZE
```

- That is, when inserting we do some probes until getting a vacant slot. H(x) can be summarized as:

`H(x, probe) =  ( h(x) + probe ) % m`

where m is the tablesize, probe is 0, 1, 2 … (until reaching a vacant slot).

Example: `m=5, h(x)= x mod m,` and inserting 5, 13, 14, 8

Double hashing

```
jumpnum = hash2(key);
while (HT[index] != NULL)
  index=(index+jumpnum)%TABLESIZE
```

Example **hash2** function:

```
hash2(key) = key%SMALLNUMBER + 1;
```

→

- `H(x, probe) = ( h(x) + probe * h2(x) ) mod m`

where i= 0, 1, 2, … (until reaching a vacant slot). Note that:

- `h2(x) != 0`,
- to be good, `h2(x)` should be co-prime with `m`,
- linear probing is just a special case of double hashing when **h2(x)=1**.

Q 5.1 a)

- You are given a hash table of size 13 and a hash function hash(key) = key % 13. Insert the following keys in the table, one-by-one, using linear probing for collision resolution:

  14, 30, 17, 55, 31, 29, 16

0	1	2	3	4	5	6	7	8	9	10	11	12

Q 5.1 b)

Keys to insert: 14, 30, 17, 55, 31, 29, 16

Now insert the same keys into an (initially empty) table of the same size (13), using double hashing for collision resolution, with hash2(key) = (key % 5) + 1

0	1	2	3	4	5	6	7	8	9	10	11	12

Quiz 1

- What is the big-O complexity to retrieve from a hash table if there are no collisions?
- A. *O(1).*
- B. O(n).
- C. O(n^2).
- D. O(*log n)*

Quiz 2

- What is the big-O complexity to insert *n* new elements into a hash table if there are no collisions?
- A. $O(1)$.
- B. $O(n)$.
- C. $O(n^2)$.
- D. $O(\log n)$

Quiz 3

- What is the disadvantage of using an array of even size as the basis for a hash table?
- **A.** Takes more space.
- **B.** Produces more collisions.
- **C.** Complicates hash calculation.
- **D.** Complicates item removal.

Quiz 4

- Given a hash table T with 25 slots that stores 2000 elements, the load factor α for T is
- **A** 80
- **B** 0.0125
- **C** 8000
- **D** 1.25

Quiz 5

- 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?

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(D)

Assignment 1

- **Implementation:**
  - minimal requirements:
    - correct interface, `Makefile` works, `test.sh` works
    - insert in correct place
    - search correctly and find all dupl, report number of comparisons
  - using memory efficiently
  - no memory leaks
- **Report:**
- https://services.unimelb.edu.au/__data/assets/pdf_file/0009/471276/Writing_Science_Laboratory_Reports_Update_051112.pdf
  - concise
  - clear
  - has proper introduction, analysis (experiment finding, discussion, compare with theory), and conclusion