COMP2521 Week 5 Tutorial

Hashing, Heaps and Tries

Jeremy Le

2026T0

Announcements

- Complete your myExperience survey of your experience of your tutors
  - Hint at 60% participation and every 10% beyond that
- Lab04 handmarking due this week!
- Exam on Monday 9th Feb

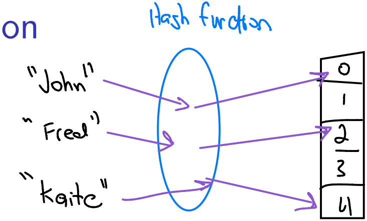
Hashing

Hash Table

- A hash table is data structure that implements an associative array, also called a dictionary or simply map.
- An associative array is an ADT that maps keys to value.
- A hash table uses a hash function to compute an index, into an array of buckets from which the desired value can be found.

- A good hash table implementation has an $O(1)$ **average** performance for insertion, lookup and deletion.
- IMPORTANT: However hash tables will always have a $O(n)$ **worst** case performance.

Hash Function



Hash Function

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The values returned by a hash function are called hash values, hash codes or simply hashes.

Hash Function - Example

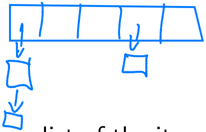
Question 1

Consider this potential hash function:

```c
int hash(char *key, int N) {
    int h = 0;
    char *c;
    for (c = key; *c != '\0'; c++) {
        h = h + (*c * i);
    }
    return h % N;
}
```

- How does this function convert strings to ints? adds ASCII values of char in string
- What are the deficiencies with this function and how can it be improved? "abc" "cba"

How to deal with hash collisions?



- Separate chaining
  - Each array slot contains a list of the items hashed to that index
  - Allows multiple items in one slot
- Linear probing
  - Check rest of array slots consecutively until an empty slot is found
- Double hashing
  - Instead of checking slots consecutively, use an increment which is determined by a secondary hash

Hash Collisions - Example

Question 3

Consider a very small hash table with only 11 entries, and a simple hash function $h(x) = x\%11$. If we start with an empty table and insert the following values in the order shown

$11 \ 16 \ 27 \ 35 \ 22 \ 20 \ 15 \ 24 \ 29 \ 19 \ 13$

give the final state of the table for each of the following collision resolution strategies

(a) separate chaining, with chains in ascending order

(b) linear probing

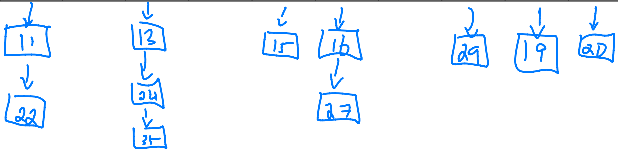
(c) double hashing, with $h_2(x) = (x\%3) + 1$

Hash Collisions - Example Working

	Key	11	16	27	35	22	20	15	24	29	19	13
x%11	$h(x)$	0	5	5	2	0	9	4	2	7	8	2
(x%3)+1	$h_2(x)$	3	2	1	3	2	3	1	1	3	2	2

a)

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]



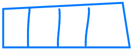
b) Linear probing

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
11	22	35	24	15	16	27	29	19	20	13

c)

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
11	13	35	24	22	16	27	15	19	20	29

Hash Collision - Example



Question 6

Fun fact: 2521 is a prime number! If a hash table which stores integer keys had 2521 slots and used double hashing for collision resolution, define a suitable secondary hash function h_2 for the table. Briefly comment on what makes it suitable.

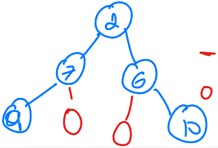
$h_1(x) = x \% 2521$

$h_2(x) = (x \% 2521) + 1$

increment to be coprime to the table size.

Heaps

$2^3 - 1$



↓

-	2	7	6	9	-	-	10
0	1	2	3	4	5	6	7

Heaps

A heap is a tree-based data structure that satisfies the heap property.

- In a *max heap*, for a given node C, if P is the parent node of C, then the key (the value) of P is greater than or equal to the key of C.
- In a *min heap*, the key of P is less than or equal to the key of C.
- The node at the "top" of the heap (with no parents) is called the root node.

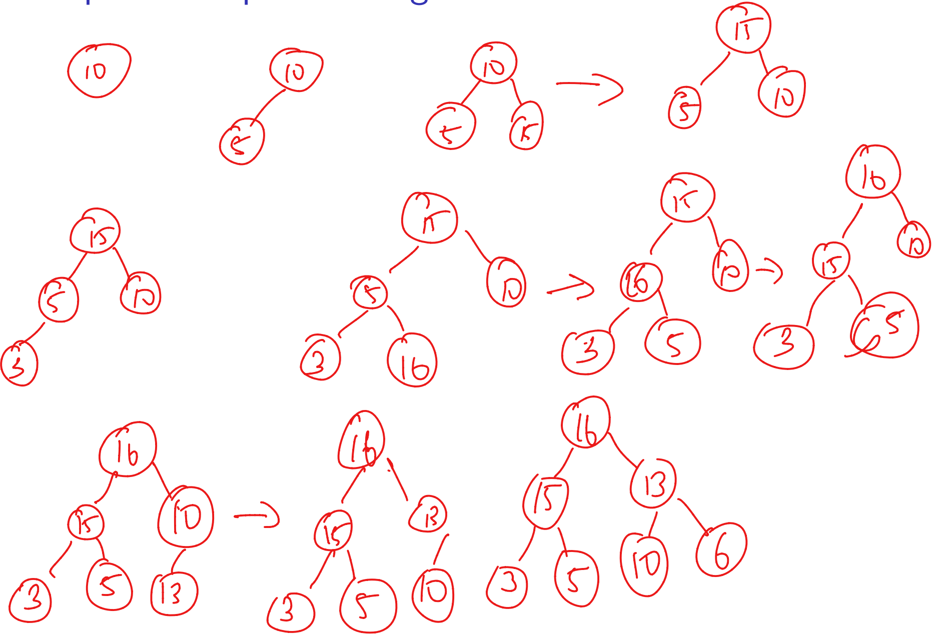
Heap - Example

Question 2

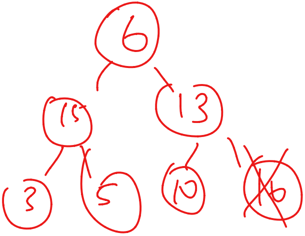
Consider the series of heap operations below. Assuming that higher values have higher priority, show the state of the heap (in both binary tree form and array form) after each operation.

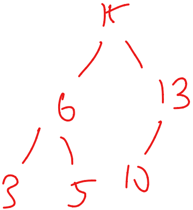
```
Heap h;
Item it;
h = heapNew();

heapInsert(h, 10);
heapInsert(h,  5);
heapInsert(h, 15);
heapInsert(h , 3);
heapInsert(h, 16);
heapInsert(h, 13);
heapInsert(h,  6);
it = heapDelete(h);
heapInsert(h,  2);
it = heapDelete(h);
it = heapDelete(h);
it = heapDelete(h);
it = heapDelete(h);
it = heapDelete(h);
```

Heaps - Example Working



Heaps - Example Working





Tries

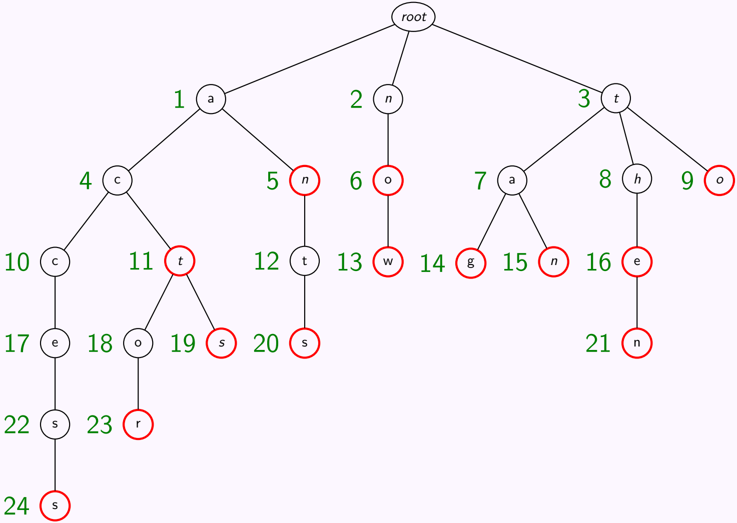
Tries

A trie is a specialised search tree data structure used to store and retrieve strings from a dictionary or a set.

Tries - Example (Searching)

Question 1

Consider the following trie:



Tries - Example (Searching)

Question 1 (continued)

Show which nodes would be visited when searching for each of the following words:

(a) ant 1 -> 5 -> 12 not a word

(b) actor 1 -> 4 -> 11 -> 18 -> 23

(c) actors

(d) a

(e) tang

Tries - Example (Insertion)

Question 2

Under what circumstances would inserting a new word into a trie not result in adding any new nodes? What would be the effect on the tree of inserting the word?

word being added is a prefix of existing word

change ending letter of word to finishing node

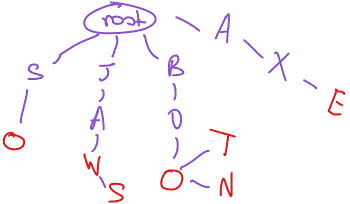
Tries - Example (Insertion)

Question 3

Show the final trie resulting from inserting the following words into an initially empty trie.

SO BOO JAWS BOON BOOT AXE JAW BOOTS SORE

Does the order that the words are inserted affect the shape of the tree?



Tries - Example (Deletion)

Question 4

What are the two cases when deleting a word from a trie? How is each case handled?

1. Deleting an internal node

2. Deleting a leaf

Thank you for the term!

Good luck on the exam!