

# COMP20007 Workshop Week 11

## Preparation:

- have *draft papers and pen ready*
- open `ws11.pptx/pdf` from [github.com/anhvir/c207](https://github.com/anhvir/c207)
- open `a2_spec.pdf`, `wokshop11.pdf` (from LMS), and
- download assignment 2 files from LMS

Counting & Radix sort: Problems T1, T2, T3

Horspool's Algorithm: Problems T4, T5, T6

Assignment 2

assignment 2

LAB

# Counting Sort for array {5,2,3,2,1,0,3,1}

# Counting Sort for sorting array A[0..n-1]: Review

*Conditions:*

small range of keys, for example,  $A[i] \leq k$ , and  $k = O(n)$

*The algorithm:*

- Build frequency array  $F[0..k]$  such that  $F[i] =$  frequency of key value  $i$
- Convert  $F[0..k]$  so that  $F[i] =$  starting index of key value  $i$  in the *sorted* array
- Using another array  $B[0..n-1]$ , scan  $A[]$  again and copy to  $B$  using:

```
k = A[i]; // here C used for convenience  
B[F[k]++] = k;
```

*Notes:*

- Time complexity:  $\Theta(n+k)$ , or  $\Theta(n)$  if  $k$  small
- Not in-place, additional memory:  $\Theta(n+k)$ , or  $\Theta(n)$  if  $k$  small

# Counting Sort & Radix Sort Exercises

1. Counting Sort Use counting sort to sort the following array of characters:

[a, b, a, a, c, d, a, a, f, c, b]

How much space is required if the array has  $n$  characters and our alphabet has  $k$  possible letters?

2. Radix Sort Use radix sort to sort the following strings:

abc bab cba ccc bbb aac abb bac bcc cab aba

As a reminder radix sort works on strings of length  $k$  by doing  $k$  passes of some other (stable) sorting algorithm, each pass sorting by the next most significant element in the string. For example in this case you would first sort by the 3rd character, then the 2nd character and then the 1st character.

3. Stable Counting Sort Which property is required to use counting sort to sort an array of tuples by only the first element, leaving the original order for tuples with the same first element. For example the input may be:

(8, campbell), (6, tal), (3, keir), ... (6, gus), (0, nick), (8, tom)

Discuss how you would ensure that counting sort satisfies this property. Can you achieve this using only arrays? How about using auxiliary linked data structures?

## Counting Sort: Problem T1

Counting Sort Use counting sort to sort the following array of characters:

[ a, b, a, a, c, d, a, a, f, c, b ]

How much space is required if the array has  $n$  characters and our alphabet has  $k$  possible letters.

## Radix Sort: Problem T2

**Radix Sort:** Use radix sort to sort the following strings:

abc bab cba ccc bbb aac abb bac bcc cab aba

As a reminder radix sort works on strings of length k by doing k passes of some other (stable) sorting algorithm, each pass sorting by the next most significant element in the string. For example in this case you would first sort by the 3rd character, then the 2nd character and then the 1st character.

## Counting Sort: Problem T3

**Stable Counting Sort:** Which property is required to use counting sort to sort an array of tuples by only the first element, leaving the original order for tuples with the same first element. For example the input may be:

(8, campbell), (6, tal), (3, keir), . . . (6, gus), (0, nick), (8, tom)

Discuss how you would ensure that counting sort satisfies this property. Can you achieve this using only arrays? How about using auxiliarry linked data structures?

# Horspool's Algorithm

**The task:** Seaching for a pattern **P** (such as “HELL” that has length  $m=5$ ) in a text **T** (such as “SHE SELLS SEA SHELLS”, having length  $n=20$ ).

**The Algorithm:**

*Stage 1:* build **SHIFT[ $x$ ]** for every possible character **x**, by:

$P = \text{HELL} \rightarrow \text{SHIFT}(L) = \text{SHIFT}(E) = \text{SHIFT}(H) =$   
 $\text{SHIFT}(\text{any\_other}) =$

*Stage 2:* searching

$i = 0 \quad 4 \quad 8 \quad 12 \quad 16 \quad 20$

**SHE SELLS SEA SHELLS**

SELL

no match,  $c = ' '$

# Horspool's Algorithm Review

**The task:** Seaching for a pattern  $P$  (such as “HELL” that has length  $m=5$ ) in a text  $T$  (such as “SHE SELLS SEA SHELLS”, having length  $n=20$ ).

## The Algorithm:

*Stage 1:* build  $\text{SHIFT}[x]$  for every possible character  $x$ , by:

1. first, set  $\text{SHIFT}[x] = m$  for all  $x$ , then
2. for each character  $x$  in  $P$ , except for the last one:  $\text{SHIFT}(x) =$  distance from the last character to last appearance of  $x$

*Stage 2:* searching, by first set  $i=m-1$ , then

1. set  $c = P[i]$ , align  $P$  with  $T$  so that  $P[m-1]$  aligned with  $T[i]$ ;
2. compare character backwardly from the last character of  $P$  until the start or until finding the first mismatch:
3. if no mismatch found: return solution which is  $i-m+1$
4. otherwise, set  $i = i + \text{SHIFT}[c]$ , back to step 1 (note: use  $c$ )

## Horspool's Algorithm: Problem T4

Use Horspool's algorithm to search for the pattern **GORE** in the string **ALGORITHM**.

## Horspool's Algorithm: Problem T5

How many character comparisons will be made by Hor-spool's algorithm in searching for each of the following patterns it the binary text of one million zeros?

(a) 01001

(b) 00010

(c) 01111

## Horspool's Algorithm: Problem T6

**Horspool's Worst-Case Time Complexity:** Using Horspool's method to search in a text of length  $n$  for a pattern of length  $m$ , what does a worst-case example look like?



## Assignment 2 (for self-understanding)

- Due: 7PM Thursday 04/June
- Marks:
  - 14 for programming ( $6/a_1 + 7/a_2 + 1$ )
  - 6 for writing ( $3/a_3+3/a_4$ )

## Programming 2: How to read data

- You can use fgets as employed in lab files week 10 (ie. week 9 in LMS.Modules)
- Example of correct read data:

```
3 5      scanf("%d %d ", &n, &x);
abc      // the trailing space in the format
cdefg    // string is IMPORTANT
agh      char s[258];
         for (i=0; i<n; i++) {
             if (!fgets(s, 256, stdin)) break;
             int l= strlen(s);
             s[l-1]= '\0'; // get rid of last \n
             // copy s to store somewhere
         }
n= i;
```

# Programming 2.1

Note: Many things will be used in both question 1a) and 1b). Make sure to organise them into functions. Example:

- reading a set of n strings
- mapping a letter/digit to an int
- compute hash value for a string
- ...?

Need to decide:

- what's the value of the hashtable's cells
- what's the SENTINEL for hashtable's cells

## Programming 2.1: The Horner's Rule

if  $S = c_0c_1 \cdots c_{\ell-1}$  is a string of  $\ell$  characters then the hash value

$$h_M(S) = \sum_{i=0}^{\ell-1} \text{chr}(c_i) \times 64^{\ell-1-i} \mod M$$

The Horner's rule:

$$h = 0$$

$$h = (h * 64 + c_0) \mod M$$

$$h = (h * 64 + c_1) \mod M$$

$$h = (h * 64 + c_2) \mod M$$

...

$$h = (h * 64 + c_{l-1}) \mod M$$

## 2.1b) Notes

Collisions should be handled using linear probing with a step size of K.

How do we know that collision cannot be solved?

When a collision cannot be solved:

- double the size of the hash table, ie.  $M = 2M$ , then build new hash table with new size M
- rehash the strings already in the old hash table, these strings must be rehashed *in the order in which they appear in the old hash table*
- continue to hash the remaining strings

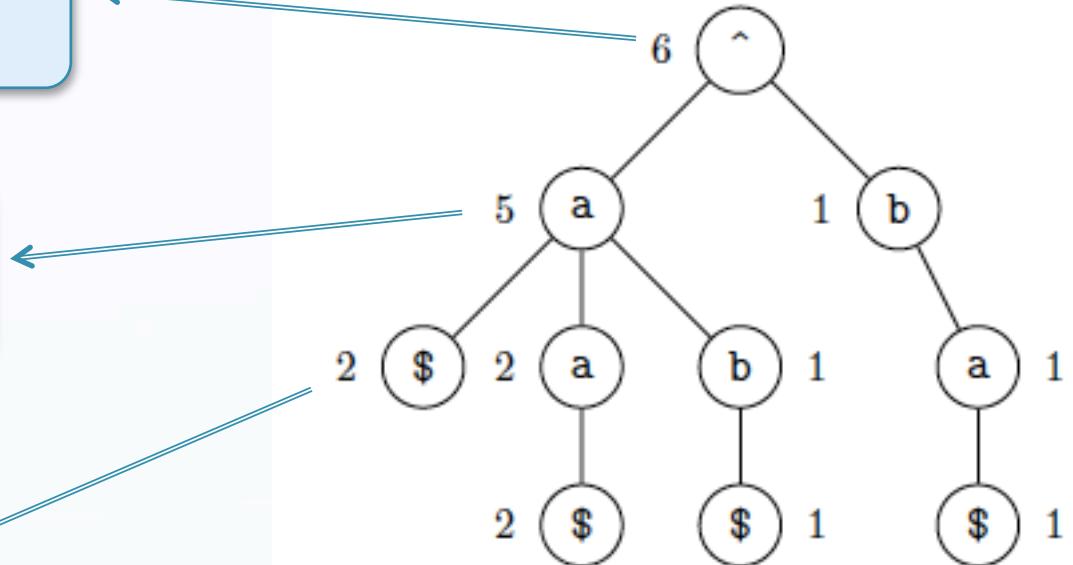


## A2.2: The Described Trie

this trie contains  
6 strings

there are 5 strings  
that share the  
prefix “a”

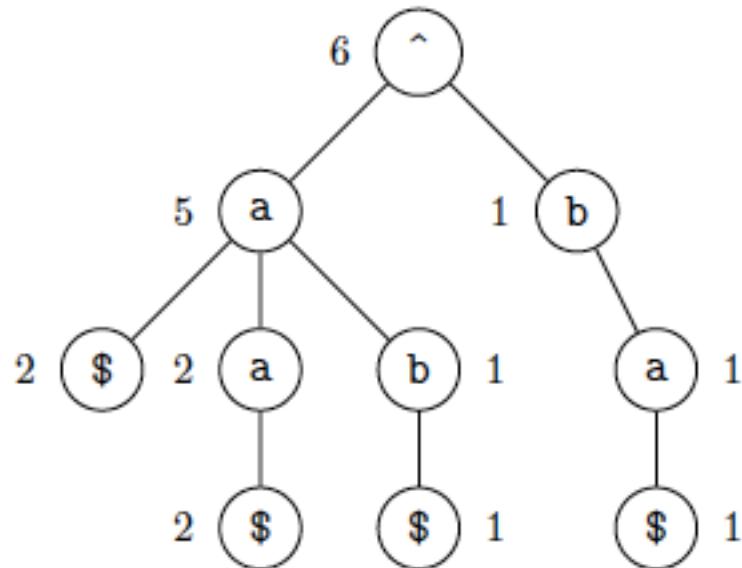
there are  
2 strings “a”



*Decide & Draw:*

- *What does the **empty trie** look like?*
- *What the empty trie become after inserting just “a”?*
- *Can we insert an empty string into the trie?*

## A2.2: How to represent?



*Operations:*

- insert a string like “abc”
- search for string “abc” and get its frequency
- search for suffix “abc” and get its frequency
- ...

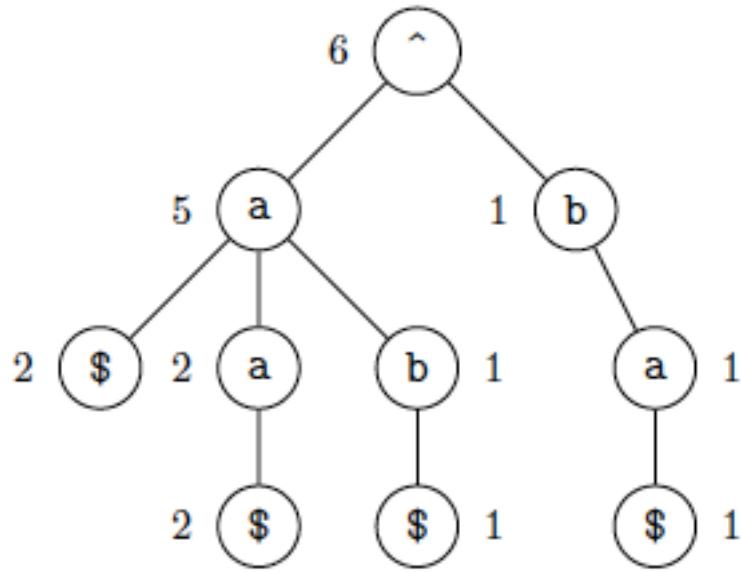
How to make difference between suffix and strings?

What a node should contains:

- a character itself?
- a frequency?
- pointers to children?

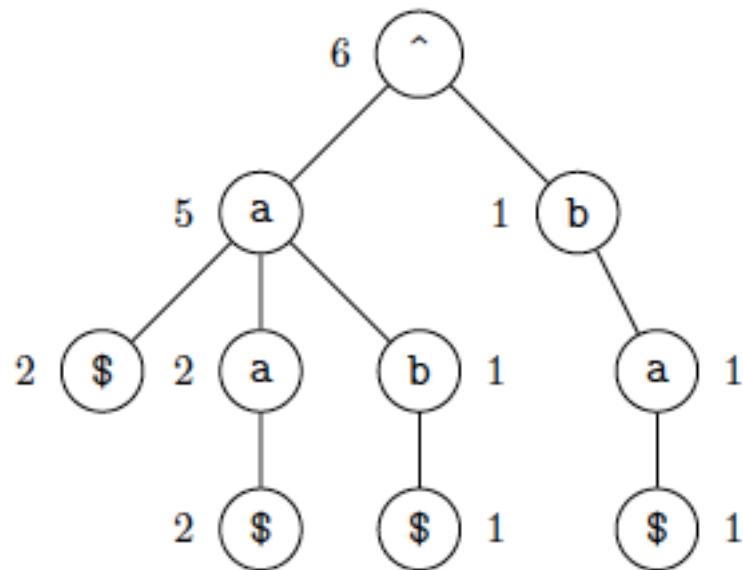
...?

## A2.2: How to insert?



*For the above trie, how to:*  
- *insert "abc" ?*  
- *insert "b" ?*

## A2.2a: What to print?

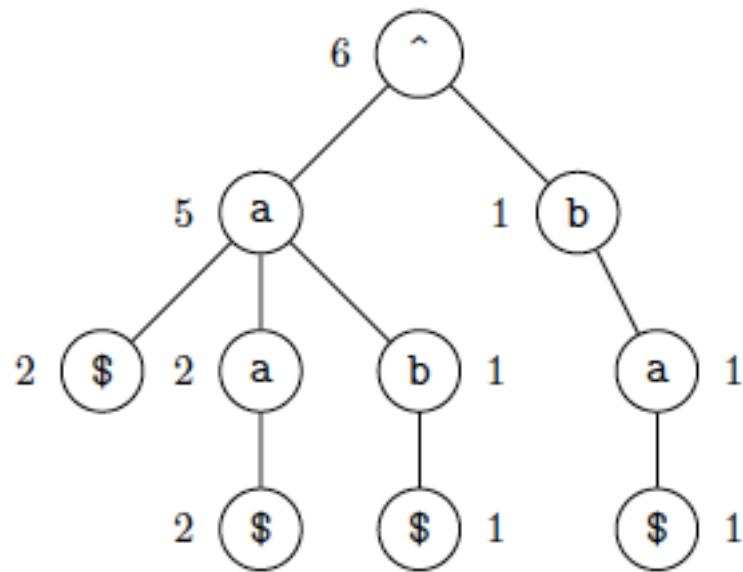


For the given trie:

^  
a  
\$  
a  
\$  
b  
\$  
b  
a  
\$

- Check that the printout is the list of nodes in pre-order traversal
- How to implement that pre-order traversal?

## A2.2b: How to print suffixes at level k?



For the given trie and  $k=2$ :

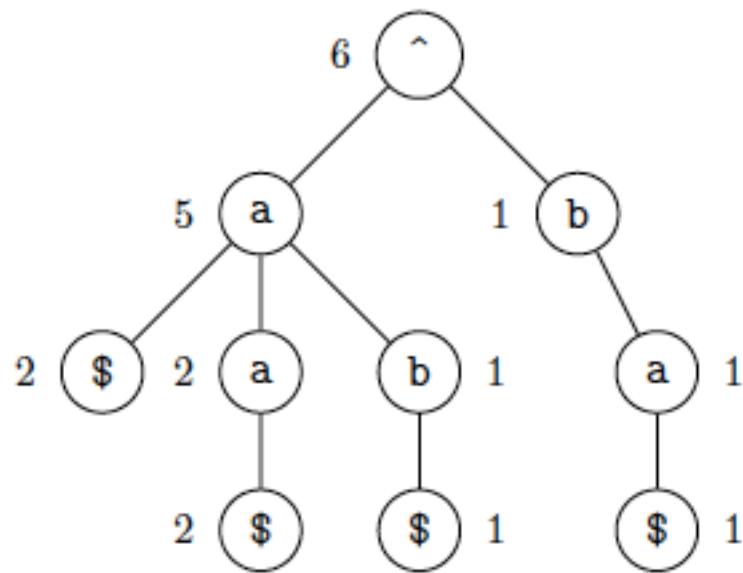
aa 5

ab 1

ba 1

- Should we use (incomplete) BFS (level-order traversal)?
- Or, can we employ (incomplete) pre-order traversal?
- At the second ‘a’, how do we generate “aa”, then, how do we generate “ab” after that?

## A2.2c: and the probability for autocompletion?



*After typing (the suffix) "a" the typist wants to finish the word as:*

- "a" – with probability  $2/5$
- "aa" – with probability  $2/5$
- "ab" – with probability  $1/5$

- How to compute the total frequencies?
- How to get all the related strings and their frequencies?
- How to print them in decreasing order of probability?

## a2.3

The task is to *select a good sorting algorithm* for each case and *justify your choice*, in a maximum of 2 sentences.

**Example case:** an airplane factory wants to improve the embedded system in its airplanes. The system uses a range of sensors that constantly collect large amounts of data from wind currents outside the airplane. This data needs to be sorted as a preprocessing step to ease their visualisation by the pilot. The sorting algorithm should be completely in-place, as extra memory usage should be minimised in an embedded system. The algorithm should also have good performance even in the worst case, since it should be robust to hijacking. What algorithm would you use?

Solution: Heapsort, because it is in-place and has guaranteed  $\Theta(n \log n)$  worst case performance, which makes it robust to adversarial attacks.

It's important to keep in mind that this is a subjective question and there might be more than one correct answer. Many real world scenarios do not have a single correct solution—this is why your argument about your chosen algorithm is important.

## a2.3

The task is to *select a good sorting algorithm* for each case and *justify your choice*, in a maximum of 2 sentences.

**Example case:** an airplane factory wants to improve the embedded system in its airplanes. The system uses a range of sensors that *constantly collect large amounts of data* from wind currents outside the airplane. This data needs to be sorted as a preprocessing step to ease their visualisation by the pilot. The sorting algorithm should be completely *in-place*, as extra memory usage should be minimised in an embedded system. The algorithm should also have *good performance even in the worst case*, since it should be robust to hijacking. What algorithm would you use?

*constantly collect large amounts of data:* all data available right away

*in-place: mergesort/distribution sorts not applicable*

*good performance: quicksort, heap sort?*

*good performance even in the worst case: no quicksort!*

Solution: Heapsort, because it is in-place and has guaranteed  $\Theta(n \log n)$  worst case performance, which makes it robust to adversarial attacks.

## a2.4

The task: