

CSE 332: Data Structures and Parallelism

Section 1: WorkLists

WorkList ADT

<code>add(work)</code>	Notifies the worklist that it must handle work
<code>peek()</code>	Returns the next item to work on
<code>next()</code>	Removes and returns the next item to work on
<code>hasWork()</code>	Returns true if there's any work left and false otherwise

0. Odd Jobs

For each of the following scenarios, choose

- (1) an ADT: `Stack` or `Queue`
- (2) a data structure: `Array`, `LinkedList with front`, or `LinkedList with front and back` and give a reason for each decision.

WorkList Situations

- (a) You're designing a tool that checks code to verify that all opening brackets, braces, parentheses, ... have closing counterparts.

(b) Disneyland has hired you to find a way to improve the processing efficiency of their long lines at attractions. There is no way to forecast how long the lines will be.

(c) A sandwich shop wants to serve customers in the order that they arrived, but also wants to look ahead to know what people have ordered (ej. 2nd person, 3rd person, ..., last person in line).

1. Trie to Delete 0's and 1's?

Suppose we inserted all possible binary strings of length 0-3 (ej. 1, 0, 10, ..., 110, 111) into a `Trie`.

(a) If we deleted all binary numbers of length 2, how many nodes would we have to delete?

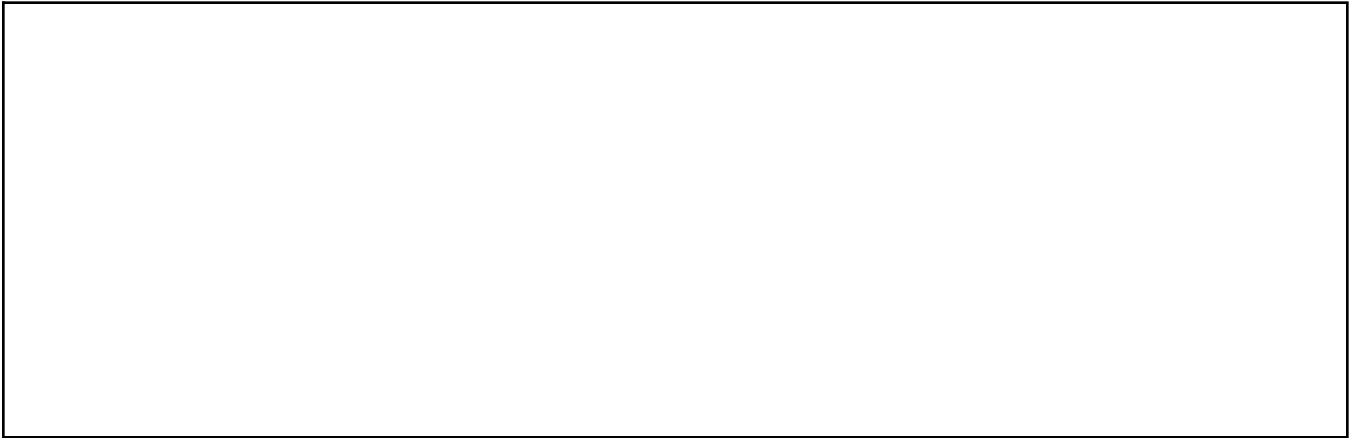
(b) After part a, if we deleted all binary numbers of length 3, how many nodes would we have to delete?

2. Call Me Maybe

(a) Suppose you want to transfer someone's phone book to a data structure so that you can call all the phone numbers with a particular area code efficiently. What data structure would you use? How would you implement it?

(b) What is the time complexity of your solution?

(c) What is the space complexity?

A large, empty rectangular box with a thin black border, intended for the user to write their answer to the question above.

3. Let's Trie to be Old School

Text on nine keys (T9)'s objective is to make it easier to type text messages with 9 keys. It allows words to be entered by a single keypress for each letter in which several letters are associated with each key. It combines the groups of letters on each phone key with a fast-access dictionary of words. It looks up in the dictionary all words corresponding to the sequence of keypresses and orders them by frequency of use. So for example, the input '2665' could be the words {book, cook, cool}. Describe how you would implement a T9 dictionary for a mobile phone.



T9 Example