CS 61B Fall 2020 — Heaps and Tries

Exam Prep Discussion 9: October 19, 2020

1 Fill in the Blanks

Fill in the following blanks related to min-heaps:

1. `removeMin` has a best case runtime of ________________ and a worst case runtime of ________________.
2. `insert` has a best case runtime of ________________ and a worst case runtime of ________________.
3. A ________________ or ________________ traversal on a min-heap can output the elements in sorted order.
4. The fourth smallest element in a min-heap with 1000 distinct elements can appear in ________________ places in the heap.
5. Given a min-heap with $2^n - 1$ distinct elements, for an element
   - to be on the second level it must be less than ________________ element(s) and greater than ________________ element(s).
   - to be on the bottommost level it must be less than ________________ element(s) and greater than ________________ element(s).

Solution:

1. `removeMin` has a best case runtime of $\Theta(1)$ and a worst case runtime of $\Theta(logN)$.
2. `insert` has a best case runtime of $\Theta(1)$ and a worst case runtime of $\Theta(logN)$.
3. A pre order or level order traversal on a min-heap can output the elements in sorted order.
4. The fourth smallest element in a min-heap with 1000 distinct elements can appear in 14 places in the heap. (it can be on the second, third, or fourth levels)
5. Given a min-heap with $2^n - 1$ distinct elements, for an element -
   - to be on the second level it must be less than $2^{(n-1)} - 2$ element(s) and greater than 1 element(s). (must be greater than the topmost and less than the elements in its subtree)
   - to be on the bottommost level it must be less than 0 element(s) and greater than n-1 element(s). (must be greater than the elements on its branch)

2 Heap Mystery

We are given the following array representing a min-heap where each letter represents a **unique** number. Assume the root of the min-heap is at index zero, i.e. `A` is the root.

Array: [A, B, C, D, E, F, G]

Four unknown operations are then executed on the min-heap. An operation is either a `removeMin` or an `insert`. The resulting state of the min-heap is shown below.

Array: [A, E, B, D, X, F, G]

(a) Determine the operations executed and their appropriate order. The first operation has already been filled in for you!

1. `removeMin()`
2. ____
3. ____
4. ____

Solution:

1. `removeMin()`
2. `insert(X)`
3. `removeMin()`
4. `insert(A)`

Explanation: We know immediately that A was removed. Then, after looking at the final state of the min-heap, we see that C was removed. Then, for A to remain in the min-heap, we see that A must have been inserted afterwards. And, after seeing a new value X in the min-heap, we see that X must have been inserted as well. We just need to determine the relative ordering of the `insert(X)` in between the operations `removeMin()` and `insert(A)`, and we see that the `insert(X)` must go before both.

(b) Fill in the following comparisons with either $>$, $<$, or ? if unknown. Note that this question does not assume a specific ordering of operations from the previous part, i.e. we don't know which of the two possible

1. X ______ D
2. X ______ C
3. B ______ C
4. G ______ X

Solution:

1. X ? D

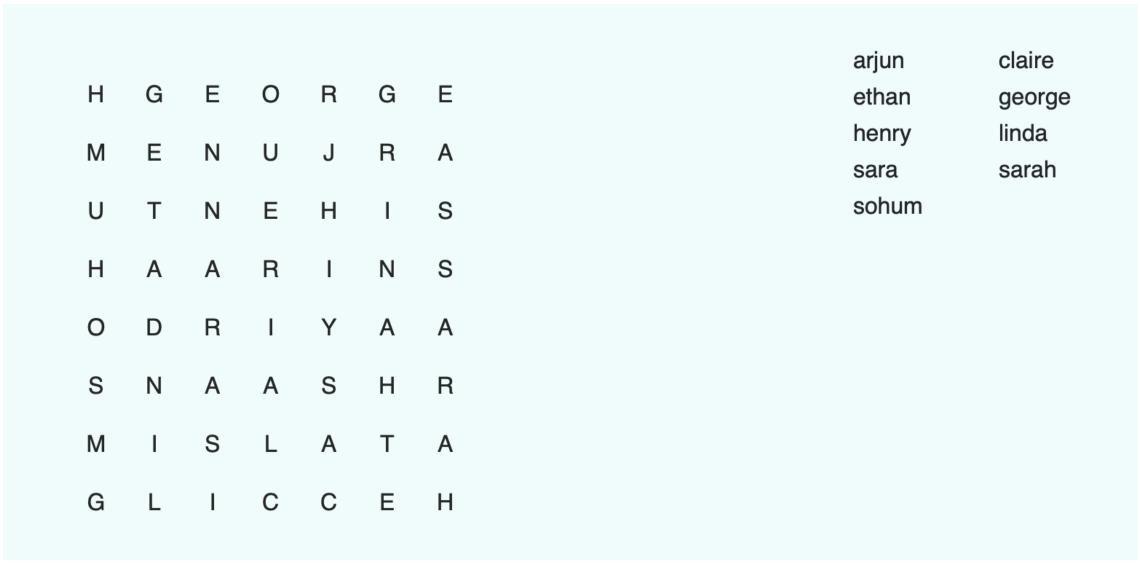
2. X $>$ C
3. B $>$ C
4. G $<$ X

Reasoning:

1. X is never compared to D
2. X must be greater than C since C is removed after X's insertion.
3. B must also be greater than C otherwise the second call to `removeMin` would have removed B
4. X must be greater than G so that it can be "promoted" to the top after the removal of C. It needs to be promoted to the top to land in its new position.

3 A Wordsearch

Given an N by N wordsearch and N words, devise an algorithm to solve the word-search in O(N^3). Each word is at most N letters. For simplicity, no word is contained within another, i.e. if the word "bear" existed, "be" could not exist as well. See below for an example wordsearch:



H	G	E	O	R	G	E
M	E	N	U	J	R	A
U	T	N	E	H	I	S
H	A	A	R	I	N	S
O	D	R	I	Y	A	A
S	N	A	A	S	H	R
M	I	S	L	A	T	A
G	L	I	C	C	E	H

arjun	claire
ethan	george
henry	linda
sara	sarah
sohum	

Hint: Add the words to a Trie, and you may find the `longestPrefixOf` operation helpful. Recall that `longestPrefixOf` accepts a `String key` and returns the longest prefix of the given string that exists in the `Trie`, or **`null`** if no prefix exists.

Algorithm: Begin by adding all the words we are querying for into a `Trie`. Next, we will iterate through each letter in the wordsearch and see if any words *start* with the given letter. For a word to start with a given letter, note that it can go in one of eight directions — N, NE, E, SE, S, SW, W, NW.

Looking at each direction, we will check if the string going in that direction has a prefix that exists in our `Trie`, which we can do using `longestPrefixOf`. Note that words are not nested inside of others, so *at most* one word can start from a given letter in a given direction. As such, if `longestPrefixOf` returns a word, we know it is the only word that goes in that direction from that letter.

For instance, if we are at the letter "H" in the top left corner of the wordsearch above and are considering the direction "SE", we would want to see if the string "HENRYHA" has a prefix that exists in the given wordsearch. To efficiently perform this query, we call `longestPrefixOf("HENRYHA")`, which, in this case, returns `"HENRY"`, and we proceed by removing `"HENRY"` from our `Trie` to signal that we found the word `"HENRY"`.

We will repeat this process until the all the words have been found, i.e. when the `Trie` is empty. Finally, note that this is a very open ended problem, so this is one of *many* possible solutions.

Runtime: We look at N^2 letters; at each letter, we execute eight calls to `longestPrefixOf` which runs in time linear to the length of the inputted string, which can be of at most length N. Thus, if we perform N work per letter and we look at N^2 letters, the runtime is $O(N^3)$.