Data Structures and Algorithms – (COMP SCI 2C03)
Winter 2021
Tutorial - 4

Feb 22, 2021

1. Which of the symbol-table implementations in this section would you use for an application that does 10^3 put() operations and 10^6 get() operations, randomly intermixed? Justify your answer.
   Answer: For an application that does 10^3 put() operations and 10^6 get() operations I would use a binary search symbol table implementation. The application does a lot more get() than put() operations and a binary search symbol table implementation has a O(log(n)) runtime complexity for the get() operation, which is better than the O(n) runtime complexity of the sequential search symbol table implementation.

2. List 3 pros and cons of the lazy delete approach in an unordered symbol table implementation.
   Answer: If the unordered ST is implemented as an array: 1- Pros- lazy deletion will save shifting time required for deletes. It could also save time for inserting an element having the same key as an existing element marked for delete. In which case, the existing item in the array can be unmarked and its associated value updated.. 2- Cons- deleted items are unused memory and waste of space (memory leakage). 3- Cons- search time is increased because of the unnecessary tracing time of the deleted items. 4- Cons - The insert time for elements never added before to the array also increases, due to the increased search time incurred due to the nodes marked for delete.

   Lazy delete in a linked list implementation of an unordered ST has the same cons listed above. The only advantage of marking an element for delete in this case, is that if an item with the same key (as the item

marked as delete) has to be inserted again, we can simply unmark the item for delete and update the value associated with the key. This eliminates the need to traverse through the entire list to see if the key exists in the linked list.

3. Suppose you have the following sorted list [3, 5, 6, 8, 11, 12, 14, 15, 17, 18] and are using the binary search algorithm given on slide #11 in C3P1.pdf. Give the sequences of elements examined to find the key 8.
   Answer: 11-5-6-8

4. Explain the running time complexity of the binary search program, and prove that it is $\in O(\log n)$.
   Answer: Let $T(N)$ be the time taken to search for an element in an array of size N. During the binary search we compare the given element (key) to the middle element, and then if the key is not equal to the middle element, we either check for it in the left half of array, of size $\lfloor N/2 \rfloor$ or the right half of array of size $\lceil n/2 \rceil$. In either case, the other approximate half of the array is discarded and no longer used for the search. Since at each iteration of the while loop the array size is reduced to approximately half, the while loop executes for $\approx \log_2 N$ times. Furthermore, at each iteration at most 5 statements are executed (infact at most 4, as when the else statement is executed the function exits). The two statements outside the while loop take time 3. Hence, the running time of the algorithm is $T(n) = 5 \log_2 n + 3$. To prove that $T(n) \in O(\log n)$, we take $c_1 = 8$ and $n_0 = 1$.

5. Write the binary search algorithm for a sorted array in decreasing order.
   Answer: For, the binary search algorithm for a sorted array in decreasing order, when the key to be searched is compared to the middle element, and if the key is smaller than the middle element, the algorithm needs to look for the key in the right sublist; otherwise it needs to look in the left sublist. Consider the binary search implementation on Slide.no 11 of C3P1.pdf. In that code, we simply need to set `lo= mid+1` and `hi = mid-1` in the `if` and `elseif` statements. Figure 1, gives the changes needed on Slide 11 of C3P1.pdf to implement the binary search algorithm for a sorted array in decreasing order.

6. Does binary search have an effect on its running time when the array

```
private int rank(Key key)                    number of keys < key
{
   int lo = 0, hi = N-1;
   while (lo <= hi)
   {
       int mid = lo + (hi - lo) / 2;
       int cmp = key.compareTo(keys[mid]);
       if      (cmp  < 0) hi = mid - 1; lo = mid + 1
       else if (cmp  > 0) lo = mid + 1; hi = mid - 1
       else if (cmp == 0) return mid;
  }
  return lo;
}
```

Figure 1: Binary search for sorted array in decreasing order

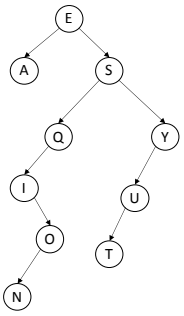
consists of duplicates?
Answer: If the element to be searched is in the array, and it has duplicates then it could be found sooner. However, if the element to be searched is not in the array then duplicates have no effect on the running time of binary search.

7. Suppose you have an array A of length 1000 and its elements are in the range [1..100]. Clearly, it must have 900 duplicates. How would you perform efficient search on this array? The array elements are fixed and you can copy elements of A to another array for efficiency.
Answer: Since the elements range from 1 to 100, we create a new smaller array A' of size 100 and initialize it to zero. Then, we scan through A to fill A' such that, if $A'[A[i]] \neq 0$, then $A'[A[i]] = A[i]$. After which all subsequent searches on A, can be performed on A'.

8. Local minimum of an array. Write a program that, given an array $a[]$ of N distinct integers, finds a local minimum: an index i such that $a[i] < a[i-1]$ and $a[i] < a[i+1]$. Your program should use approximately $2\log_2 N$ compares in the worst case.
Answer : Examine the middle value $a[N/2]$ and its two neighbours $a[N/2-1]$ and $a[N/2+1]$. If $a[N/2]$ is a local minimum, stop; otherwise search in the half with the smaller neighbour.

9. Draw the BST that results when you insert the keys E A S Y Q U E S T I O N, in that order (associating the value i with the i-th key, as per the convention in the text) into an initially empty tree. How many

compares are needed to build the tree?



Answer: Sum of comparisons: 28

Please provide a breakdown of the 28 compares.

10. For the set of $\{1, 4, 5, 10, 16, 17, 21\}$ of keys, draw binary search trees of heights 2, 3, 4, 5, and 6.

Answer: