Question Bank
Data Structures and Algorithms, 2020
Version_Date: 22nd September, 2020

Topic: Asymptotic analysis

1. The number of operations executed by algorithms A and B is $8n\ logn$ and $2n^2$, respectively. Determine n_0 such that A is better than B for $n \geq n_0$.

2. The number of operations executed by algorithms A and B is $40n^2$ and $2n^3$, respectively. Determine n_0 such that A is better than B for $n \geq n_0$.

3. Order the following functions by asymptotic growth rate.

$4n\ logn + 2n$	2^{10}	2^{logn}
$3n + 100logn$	$4n$	2^n
$n^2 + 10n$	n^3	$nlogn$

4. Suppose you have algorithms with the five running times listed below. (Assume these are the exact running times.) How much slower do each of these algorithms get when you (a) double the input size, or (b) increase the input size by one?

   (a) n^2
   (b) n^3
   (c) $100n^2$
   (d) $nlogn$
   (e) 2^n

5. In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

	$f(n)$	$g(n)$
(a)	$n - 100$	$n - 200$
(b)	$100n + logn$	$n + (logn)^2$
(c)	$log2n$	$log\ 3n$
(d)	$n^{1.01}$	$nlog^2n$
(e)	$n^{0.1}$	$(log\ n)^{10}$
(f)	$\sqrt{n}$	$(log\ n)^3$
(g)	$n2^n$	3^n
(h)	$n!$	2^n

6. Describe an efficient algorithm for finding the ten largest elements in a sequence of size n. What is the running time of your algorithm?

7. Show that for any real constant a and b, where $b > 0$,
$$(n + a)^b = \Theta(n^b)$$

8. Show that, if c is a positive real number, then $g(n) = 1 + c + c^2 + \ldots + c^n$ is:

   (a) $\Theta(1)$ *if* $c < 1$.
   (b) $\Theta(n)$ *if* $c = 1$.
   (c) $\Theta(c^n)$ *if* $c > 1$.

9. Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

   (a) $\log_2 f(n)$ *is* $O(\log_2 g(n))$.
   (b) $2^{f(n)}$ *is* $O(2^{g(n)})$
   (c) $f(n)^2$ *is* $O(g(n)^2)$.

10. Perform experimental analysis to test the hypothesis that Python's sorted method runs in $O(n \log n)$ time on average.

Topic: Divide & Conquer

1. Use the divide and conquer integer multiplication algorithm to multiply the two binary integers 10011011 and 10111010.

2. You are given a unimodal array of n distinct elements, meaning that its entries are in increasing order up until its maximum elements, after which its elements are in decreasing order. Give an algorithm to compute the maximum element of a unimodal array that runs in $O(\log n)$ time.

3. You are given a sorted (from smallest to largest) array A of n distinct integers which can be positive, negative, or zero. Design the fastest algorithm you can for deciding whether or not there is an index i such that $A[i] = i$.

4. You are given an array of n elements, and you notice that some of the elements are duplicates; that is, they appear more than once in array. Show how to remove all duplicates from the array in time $O(n \log n)$.

5. You are interested in analyzing some hard to obtain data from two separate databases. Each database contains n numerical values--so there are $2n$ values total--and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we will define here to be the n^{th} smallest value.

   However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

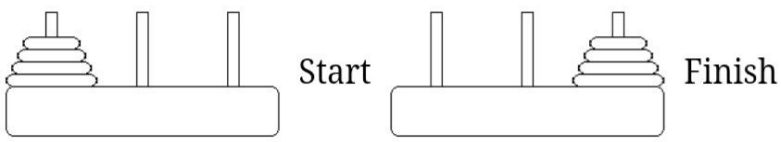
   Give an algorithm that finds the median value using at most $O(\log n)$ queries.

6. Suppose you're consulting for a bank that's concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we'll say that two bank cards are equivalent if they correspond to the same account.

   It's very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent.

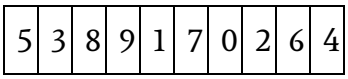
   Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them into the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.

7. Towers of Hanoi. Given a game board with three pegs and a set of disks of different diameter all stacked from smallest to largest on the leftmost peg, move all of the disks to the rightmost peg following these two rules. First, only one disk may be moved at a time. Second, a larger diameter disk may never be placed on a smaller disk. Any number of disks can be used. Implement this in Python.



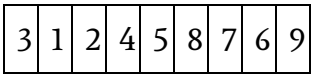
Topic: Sorting

1. Suppose we run `MergeSort` on the following input array:



   Fast forward to the moment after the two outermost recursive calls complete, but before the final `Merge` step. Thinking of the two 5-element output arrays of the recursive calls as a glued-together 10-element array, which number is in the 7th position?

2. Consider the `Partition` subroutine employed by `QuickSort`. You are told that the following array has just been partitioned around some pivot elements:



   Which of the elements could have been the pivot element? List all that apply. There could be more than one possibility.

3. Show that the running time of the merge-sort algorithm on n-element sequence is $O(n \log n)$, even when n is not a power of 2.

4. Consider a modification of the deterministic version of the quick-sort algorithm where we choose the element at index $\lfloor n/2 \rfloor$ as our pivot. Describe the kind of sequence that would cause this version of quick-sort to run in $\Omega(n^2)$ time.

5. Suppose S is a sequence of n-values, each equal to 0 or 1. How long will it take to sort S with the merge-sort algorithm? What about quick-sort?

6. Describe and analyze an efficient method for removing all duplicates from a collection A of n elements.

7. Experimentally compare the performance of in-place quick-sort and a version of quick-sort that is not in-place.

8. Given an array A of n integers in the range $[0, n^2 - 1]$, describe a simple method for sorting A in $O(n)$ time.

9. Show that quicksort's best-case running time is $\Omega(n \lg n)$.

10. Perform a series of benchmarking tests on a version of merge-sort and quick-sort to determine which one is faster. Your test should include sequences that are "random" as well as "almost" sorted.

Topic: Data Structures

1. Show how to implement a stack using two queues. Analyze the running time of the stack operations.

2. Implement a stack using a singly linked list L. The operations PUSH and POP should still take $O(1)$ time.

3. Demonstrate what happens when we insert the keys 5, 28, 19, 15, 20, 33, 12, 17, 10 into a hash table with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be $h(k) = k \bmod 9$.

4. Consider a hash table of size m = 1000 and a corresponding hash function $h(k) = \lfloor m(kA \bmod 1) \rfloor$ for $A = (\sqrt{5} - 1)/2$. Compute the locations to which the keys 61, 62, 63, 64 and 65 are mapped.

5. Consider a binary search tree T whose keys are distinct. Show that if the right subtree of a node x in T is empty and x has a successor y, then y is the lowest ancestor of x whose left child is also an ancestor of x.

6. Implement a function that reverses a list of elements by pushing them onto a stack in one order, and writing them back to the list in reversed order.

7. Describe a non-recursive algorithm for enumerating all permutations of the numbers {1, 2, ...,n} using an explicit stack.

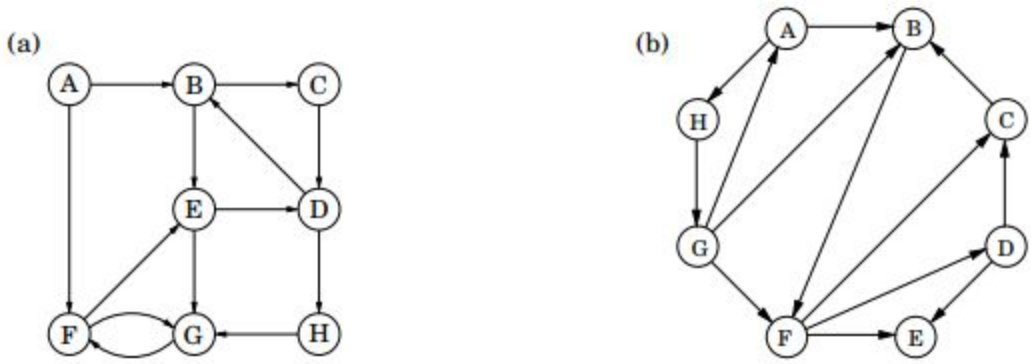
8. Given the set of element {a,b,c,d,e,f} stored in a list, show the final state of the list, assuming we use the move-to-front heuristic and access the elements according to the following sequence: (a,b,c,d,e,f,a,c,f,b,d,e).

9. Show that any n-node binary tree can be converted to any other n-node binary tree using $O(n)$ rotations.

10. Implement the dictionary operations INSERT, DELETE, and SEARCH using singly linked, circular lists. What are the running times of your procedures?

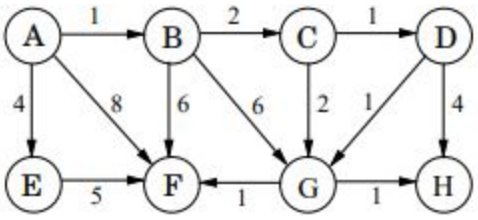
Topic: Graph algorithms

1. Perform depth-first search on each of the following graphs; whenever there's a choice of vertices, pick the one that is alphabetically first. Classify each edge as a tree edge, forward edge, back edge, or cross edge, and give `pre` and `post` numbers of each vertex.



2. Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (it should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m+n)$ for a graph with n nodes and m edges.

3. Suppose Dijkstra's algorithm is run on the following graph, starting at node A.



   (a) Draw a table showing the intermediate distance values all the nodes at each iteration of the algorithm.
   (b) Show the final shortest-path tree.

4. Explain how a vertex u of a directed graph can end up in a depth-first tree containing only u, even though u has both incoming and outgoing edges in G.

5. Suppose that we represent the graph $G = (V,E)$ as an adjacency matrix. Give a simple implementation of Prim's algorithm for this case that runs in $O(V^2)$ time.

6. Give pseudo-code for performing the operation insert_edge(u,v,x) in $O(1)$ time using the adjacency matrix representation.

7. Draw a simple, connected, weighted graph with 8 vertices and 16 edges, each with unique edge weights. Identify one vertex as a "start" vertex and illustrate a running of Dijkstra's algorithm on this graph.

8. Design an experimental comparison of repeated DFS traversals versus the Floyd-Warshall algorithm for computing the transitive closure of a directed graph.

9. Let G be an undirected graph whose vertices are the integers 1 through 8, and let the adjacent vertices of each vertex be given by the table below:

vertex	adjacent vertices
1	(2, 3, 4)
2	(1, 3, 4)
3	(1, 2, 4)
4	(1, 2, 3, 6)
5	(6, 7, 8)
6	(4, 5, 7)
7	(5, 6, 8)
8	(5, 7)

Assume that, in a traversal of G, the adjacent vertices of a given vertex are returned in the same order as they are listed in the table above.

(a) Draw G.
(b) Given the sequence of vertices of G visited using a DFS traversal staring at vertex 1.
(c) Give sequence of vertices visited using a BFS traversal starting at vertex 1.