Exam Exercises

Time Complexity

1. Assume that each of the expressions below gives the processing time T(n) spent by an algorithm for solving a problem of size n. Select the dominant term(s) having the steepest increase in n and specify the lowest Big-Oh complexity of each algorithm

Expression	Dominant term(s)	$O(\ldots)$
$5 + 0.001n^3 + 0.025n$		
$500n + 100n^{1.5} + 50n \log_{10} n$		
$0.3n + 5n^{1.5} + 2.5 \cdot n^{1.75}$		
$n^2 \log_2 n + n(\log_2 n)^2$		
$n \log_3 n + n \log_2 n$		
$3 \log_8 n + \log_2 \log_2 \log_2 n$		
$100n + 0.01n^2$		
$0.01n + 100n^2$		
$2n + n^{0.5} + 0.5n^{1.25}$		
$0.01n \log_2 n + n(\log_2 n)^2$		
$100n \log_3 n + n^3 + 100n$		
$0.003 \log_4 n + \log_2 \log_2 n$		

2. Work out the computational complexity (in the “Big-Oh” sense) of the following piece of code and explain how you derived it using the basic features of the “Big-Oh” notation:

```
for( int bound = 1; bound <= n; bound *= 2 ) {
   for( int i = 0; i < bound; i++ ) {
      for( int j = 0; j < n; j += 2 ) {
         ... // constant number of operations
      }
      for( int j = 1; j < n; j *= 2 ) {
         ... // constant number of operations
      }
   }
}
```

3. Determine an explicit formula for the time T(n) of processing an array of size n if T(n) relates to the average of T(n − 1), . . . , T(0) as follows:

$$T(n) = \frac{2}{n}\left(T(0) + \cdots + T(n-1)\right) + c$$

where T(0) = 0.

Hint: You might need the equation $\frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + \cdots + \frac{1}{n(n+1)} = \frac{n}{n+1}$ for deriving the explicit formula for $T(n)$.

Brute-Force

4. Given an unsorted list of size n, write an algorithm to find the kth smallest value? Your algorithm must be O(n2) in the worst case. You cannot use a sorting algorithm. Describe your algorithm using pseudo-code or a precise step-by-step description.

5. Given the following algorithm and the following input, how many comparisons are made?

a.

```
char T[] = {NOBODYNOTICED}
char P[] = {NOT}

for (i =0; i < n-m; i++) {
  j = 0;
  while (j < m && P[j] == T[i+j])
    j++;
  if (j ==m)
    return i
  return -1;
}
```

Answer ______

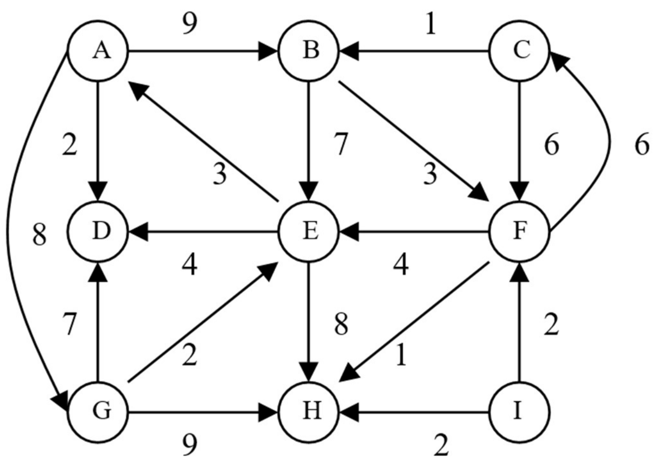
b. . Given an example of a text of length 10 and a pattern of length 5 that constitutes a worst case for the algorithm above?

Pattern = ________________________

Text = _______________

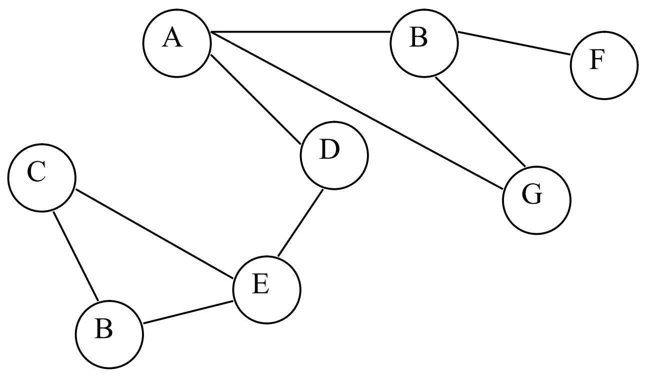
Graph-Traversal

6. Given the following graph, perform a DFS starting at node I. Break ties alphabetically.



  a. Show the DFS tree
  b. List the vertices in the order in which they were pushed on to the DFS stack
  c. List the vertices in the order in which they were popped off the DFS stack

7. Given the following graph, perform a breadth first traversal starting at node A. To break ties, try to visit vertices in alphabetical order. Mark each node with count (i.e., the BFS number).



Decrease/Divide and Conquer

8. A d-dimensional vector ⟨u1, u2,...,ud⟩ screens the vector ⟨v1, v2,...,vd⟩ iff there is a permutation π on {1, 2,...,d} such that u1 > vπ(1), u2 > vπ(2), ..., ud > vπ(d). For example, ⟨7, 9, 4⟩ screens ⟨7, 6, 3⟩ because one permutation of the latter is ⟨6, 7, 3⟩. On the other hand, ⟨7, 9, 4, 5⟩ does not screen ⟨7, 6, 3, 5⟩. The last example also shows that we can have vectors u and v, neither of which screens the other.

   Using pseudo-code, give an algorithm which determines, given two vectors u and v, whether u screens v. It should return True iff u screens v. If it helps, you an assume that vectors are arrays. Make the algorithm as efficient as you can, and state, as precisely as you can, its worst-case time complexity (and, if possible, average-case as well)

9. Consider two sets of integers, X = [x1, x2, . . . , xn] and Y = [y1, y2, . . . , yn]. Write two versions of a FindUncommon(X, Y) algorithm to find the uncommon elements in both sets. Each of your algorithms should return an array with the uncommon elements, or an empty array if there are no uncommon elements.

   You may make use of any algorithm introduced in the lectures to help you develop your solution. That is, you do not have to write the 'standard' algorithms – just use them. Therefore, you should be able to write each algorithm in about 10 lines of code.

   a. Write a pre-sorting based algorithm of FindUncommon(X, Y). Your algorithm should strictly run in O (n log n).

    b. Write a Hashing based algorithm of FindUncommon(X, Y). Your algorithm should run in O (n)

Tree Related

10. Let T be an arbitrary binary tree. Let us call a given node of T an AVL node if its balance factor (i.e., the difference between the heights of the left and right subtrees) is -1, 0 or 1. Consider the function markNonAVLNodes(T) that "marks" all nodes in T that do not satisfy the balance factor condition.
    a. Write a recursive algorithm in pseudocode for markNonAVLNodes(T).
    b. Write an iterative algorithm in pseudocode for markNonAVLNodes(T)
    c. Demonstrate the time complexity of the developed algorithms.

11. Assume the following notation/operations on AVL trees. An empty AVL tree is denoted E. A non-empty AVL tree T has three attributes:
    - The key T.key is the root node's key.
    - The left child T.left is T's left subtree, which is an AVL tree (possibly E).
    - The right child T.right is T's right subtree, which is an AVL tree (possibly E).

    a. Write a function RangeCount(T, lo, hi) to count the number of nodes in an AVL tree with root T, where the key value is in the range lo ≤ key ≤ hi. Your algorithm should run in O(n) time, assuming that n is the number of nodes in the AVL tree, T.
    b. Describe an alternative version of the RangeCount(T, lo, hi) function that runs in O(log n) time.

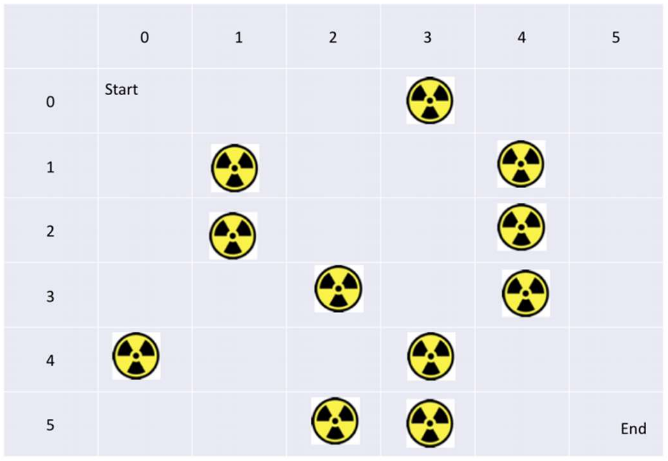
Recursion & Dynamic Programming

12. In a country popular for train travel, you have planned some train travelling one year in advance. The days of the year that you will travel is given as an array days. Each day is an integer from 1 to 365.

    Train tickets are sold in 3 different ways: a 1-day pass is sold for costs[0] dollars; a 7-day pass is sold for costs[1] dollars; a 30-day pass is sold for costs[2] dollars.
    The passes allow that many days of consecutive travel. For example, if we get a 7-day pass on day 2, then we can travel for 7 days: day 2, 3, 4, 5, 6, 7, and 8. Return the minimum number of dollars you need to travel every day in the given list of days.

13. As part of the development of the new student precinct at the University of Melbourne, an old single storey building previously used for the storage of radioactive isotopes was scheduled to be

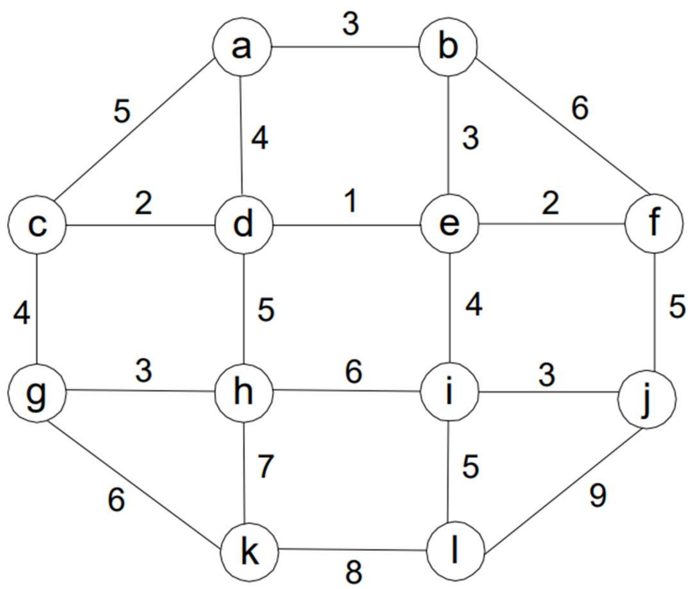
demolished. Unfortunately, higher than acceptable levels of radiation (values above a threshold level) were observed in some parts of the building. Consequently, the demolition contractors developed a n×n grid or map to record the relative radiation levels on the building floor plan. Specifically, they used a two-dimensional binary array A[0, .., n − 1][0, .., n − 1], where a value A[i][j] = 1 indicated a radiation level above the acceptable threshold level, and a value of A[i][j] = 0 indicated 'safe' locations, that is, where the recorded radiation level was below the threshold value. One of the contractors had to navigate their way across the building, starting at position A[0][0] and ending at position A[n − 1][n − 1]. Given the delicate nature of this work, the contractor could only move one cell at a time, by either moving one cell to the right or one cell down. They also wanted to minimize the number of cells that they moved through where the recorded radiation level was above the acceptable threshold level (ie., where A[i][j] = 1). Write a recursive function to determine the minimum number of cells that the contractor must pass through where the radiation level was above the acceptable threshold level. For the example below (where the radiation symbol indicates A[i][j] = 1), your algorithm should return a value of 1.



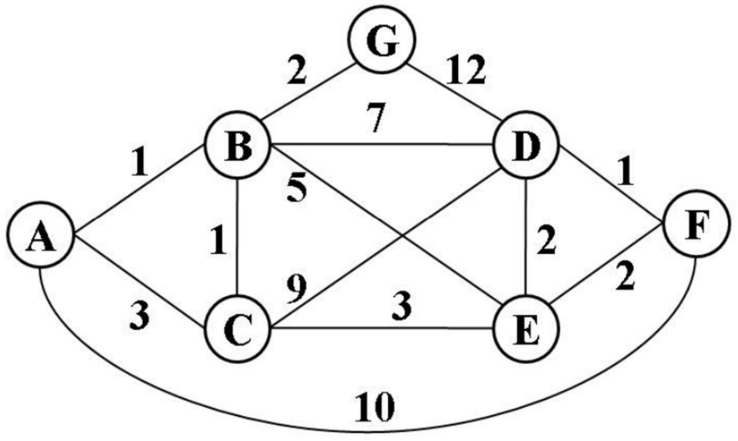
	0	1	2	3	4	5
0	Start			☢		
1		☢			☢	
2		☢			☢	
3			☢		☢	
4	☢			☢		
5			☢	☢		End

Greedy Algorithms

14. Apply Prim's algorithm to the following graph. Include in the priority queue only the fringe vertices (the vertices not in the current tree which are adjacent to at least one tree vertex).



15. Consider the following undirected, weighted graph:



Step through Dijkstra's algorithm to calculate the single-source shortest paths from A to every other vertex. Show your steps in the table below. Cross out old values and write in new ones, from left to right within each cell, as the algorithm proceeds. Also list the vertices in the order which you marked them known. Finally, indicate the lowest-cast path from node A to node F.