May 2021. Algorithms. Qualifying exam with solutions.

Closed books, closed notes.

1. (10 pts) Suppose that we have numbers between 1 and 100 in a binary search tree and want to search for the number 45. Which (possibly multiple) of the following sequences could be the sequence of nodes examined?
   (a) 5, 2, 1, 10, 39, 34, 77, 63.
   (b) 8, 7, 6, 5, 4, 3, 2, 1.
   (c) 9, 8, 63, 0, 4, 3, 2, 1.
   (d) 50, 25, 26, 27, 40, 44, 42.
   (e) 32, 48, 40, 44, 42, 43.

   Solution: For any number in the sequence, all numbers after it must be either all greater than the number or all less than or equal to the number. Possible sequences are (b), (d), (e). Sequences (a) and (c) can not be sequences of nodes examined.

2. (15 pts) Consider a binary heap (min-heap) with n nodes. Give an efficient algorithm that would find (print) all nodes in the heap that have keys less than a given number X. Analyze the running time of your algorithm.

   Solution: It can be done recursively in the following manner. Start from the root of the heap. If the value of the root is smaller than X then print this value and call the procedure recursively once for its left child and once for its right child . If the value of a node is bigger or equal than X then the procedure stops without printing that value. The complexity of this algorithm is O(n), where n is the total number of nodes in the heap. This bound takes place in the worst case, where the value of every node in the heap will be smaller than X, so the procedure has to call each node of the heap.

3. Recall that a contiguous subsequence of a sequence S is a subsequence made up of consecutive elements of S. E.g., if S is {5, 15, -30, 10, -5, 40, 10} then 15, -30, 10 is a contiguous subsequence.
   Consider the following **problem**:
   You are given as input a sequence L of n numbers, where each number is at most 100. You task is to partition it into as few contiguous subsequences as possible such that the sum of numbers within each contiguous subsequence is at most 100.
   For **example**, for the input L=[80, -40, 30, 60, 20, 30], the optimal partition has two contiguous subsequences: [80] and [-40, 30, 60, 20, 30]. Indeed, each of the numbers from [80, -40, 30, 60, 20, 30] is in exactly one of the two contiguous subsequences, and the sum of numbers in each of the two subsequences is less than or equal to 100. Notice, that the optimal partition in this example can not consist of only one subsequence because the sum of numbers in [80, -40, 30, 60, 20, 30] is greater than 100.
   Subproblems are defined as follows: Let C[i] denote the number of contiguous subsequences in the optimal solution of the sequence consisting of the first i numbers of the input sequence L.
   For instance, for the above example, C[3] = 1 (the first 3 numbers in L are 80, -40, 30, and they need only one contiguous subsequence ([80, -40, 30]) and C[6] = 2 as shown earlier.

   (a) (8 pts) Let L be a sequence of n numbers. Suppose that the last subsequence in the optimal solution for all of L has k terms. For example, in the input L=[80, -40, 30, 60, 20, 30] from the earlier example, k = 5 (the optimal

solution for L has five numbers in the last subsequence). Write a formula showing how to compute C[n] from k and from earlier values of C.
(b) (12 pts) Write a recursive solution to subproblems, that is, express C[i] in terms of smaller subproblems.

Solution:

(a) C[n] = C[n-k] + 1
(b) C[i] = $\min_{1 \le k \le i-1} (C[i-k]+1)$ if i>1
C[i] = 1 if i=1.

4. Suppose you are choosing between the following three algorithms:

- Algorithm A solves problems of size n by recursively solving one subproblem of size n-1 and then obtaining the solution to the original problem in linear time.
- Algorithm B solves problems of size n by recursively solving 16 subproblems of size n/4 and then combining the solutions in quadratic time.
- Algorithm C solves problems by dividing them into 10 subproblems of size n/3, recursively solving each subproblem, and then combining the solutions in quadratic time.

a) (24 pts; 8pts per each algorithm) Compute the running time of each of these algorithms (in big-O notation). Show your work.
b) (5 pts) Which algorithm would you choose? Explain your answer.

Solution:

a) Algorithm A. Recurrence for the running time: $T(n) = T(n-1) + cn$. Solving it using recursion tree method.

	level	argument of T
cn	0	n
|		
c(n-1)	1	n-1
|		
c(n-2)	2	n-2
.		
.		
c(n-i)	i	n-i
.		
.		
T(1)	k	n-k

Since on the last level k the argument of T is equal to 1 (T(1)), therefore, $n-k=1$ and $k=n-1$.
Total cost:

$$T(n) = cn + c(n-1) + c(n-2) + \cdots + c(n-k+1) + T(1)$$
$$= T(1) + c(n-k+1) + \cdots + c(n-2) + c(n-1) + cn$$
$$= T(1) + c(2+3+\cdots+(n-1)+n)$$
$$= T(1) + c\frac{(n+2)(n-1)}{2} = O(n^2)$$

Algorithm B. Recurrence for the running time: $T(n) = 16T(n/4) + cn^2$. Solving it using the Master theorem.

Since $\log_4(16) = 2$, the Master Theorem gives us $T(n) = O(n^2 \log n)$.

Algorithm C: Recurrence for the running time: $T(n) = 10T(n/3) + cn^2$. Solving it using the Master theorem.

Since $\log_3(10) > 2$, the Master Theorem gives us $T(n) = O(n^{\log_3 10})$.

b) We want to pick the one with the best asymptotic running time, A.

5. Figure 1 below shows a directed graph. Figure 2 shows the same graph but without directions on edges.

Figure 1

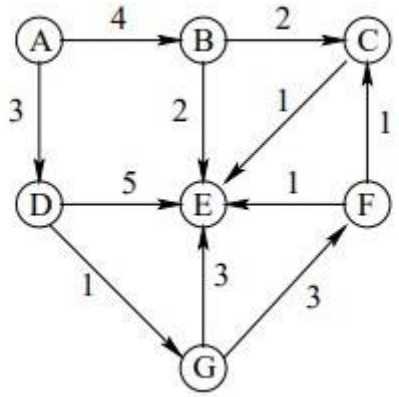
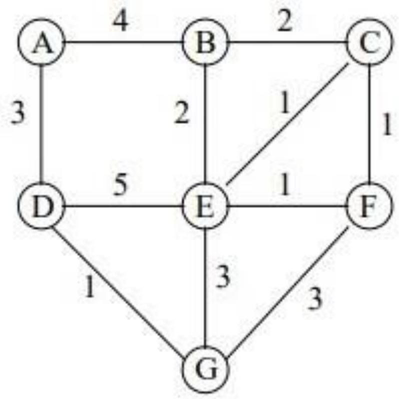


Figure 2



a) (10 points) Run Dijkstra's algorithm on the graph of Figure 1 starting at vertex A. If there are any ties, the vertex which is alphabetically first comes first. List the vertices in the order in which they are deleted from the priority queue and for each write the shortest distance from A to the vertex.

b) (8 points) Run Kruskal's algorithm on the graph in Figure 2, in case of ties add the edge which is lexicographically first. (Assume that equal weight edges are ordered lexicographically by the labels of their vertices assuming that the lower labeled vertex always comes first when specifying an edge, e.g. (C, E) is before (C, F) which in turn is before (D, G)). List the edges in the order in which they are added to the developing MST (minimum spanning tree).

c) (8 points) Run Prim's algorithm on the graph in Figure 2; whenever there is a choice of vertices, always use alphabetic ordering (e.g., start from node A). List the edges in the order in which they are added to the developing MST.

Solution:

a) A-0, D-3, B-4, G-4, C-6, E-6, F-7
b) (C,E) (C,F) (D,G) (B,C) (A,D) (E,G)
c) (A,D) (D,G) (E,G) (C,E) (C,F) (B,C)