COMP 3711 – Spring 2019
Tutorial 5

1. Your input is k sorted lists that need to be merged into one sorted list. The "obvious" solution is to modify the merging procedure from mergesort; at every step, compare the smallest items from each list and move the minimum one among them to the sorted list.

   Finding the minimum value requires $O(k)$ time so, if the lists contain n items in *total* the full k-way merge would take $O(nk)$ time.

   This can be solved faster.

   Design an $O(n \log k)$-time algorithm to merge k sorted lists into one sorted list by making use of priority queues.

   Note that each sorted list may contain a different number of elements.

2. The problem input is n/k lists:
   (i) Each list has size k, and
   (ii) for $i = 1$ to n/k, the elements in list $i - 1$ are all less than all the elements in list i

   The obvious algorithm to fully sort these items is to sort each list separately and then concatenate the sorted lists together. This uses $\frac{n}{k} O(k \log k) = O(n \log k)$ comparisons.

   Show that this is the best possible. That is, any comparison-based sorting algorithm to sort the n/k lists into one sorted list with n elements will need to make at least $\Omega(n \log k)$ comparisons.

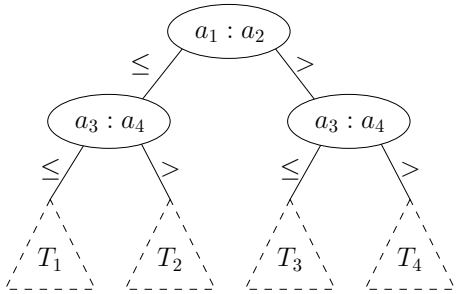
   Note that you can not derive this lower bound by simply combining the lower bounds for the individual lists.

3. In this problem, you are given n integers to sort.

   (a) You are told they are all in the range $[0, n^2 - 1]$. How fast can you sort them?

   (b) Now you are told they are all in the range $[0, n^t - 1]$ for some fixed t. How fast can you sort them?

4. The figure below shows part of the decision tree for mergesort operating on a list of 4 numbers, a_1, a_2, a_3, a_4. Please expand subtree T_3, i.e., show all the internal (comparison) nodes and leaves in subtree T_3.



5. The goal of the interval partitioning problem (i.e., the classroom assignment problem) taught in lecture was to open as few classrooms as possible to accommodate all of the classes. The greedy algorithm taught, sorted the classes by starting time.

   It then ran through the classes one at a time; at each step it first checked if there was an available empty classroom. Only if there was no such classroom would it open a new classroom.

   Prove that if the classes are sorted by finishing time the algorithm might not give a correct answer.

   Note: Proving that something does not work usually means to find a counter-example.

6. Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer and the coins have values $c_1 < c_2 < \cdots < c_k$.

   The *Greedy Algorithm* for change making is to use as many coins as possible of size c_k then as many as possible of size c_{k-1} etc.

   The setup assumes that $c_1 = 1$, to ensure that change can always be made.

   As an example, suppose that $(c_1, c_2, c_3.c_4) = (1, 5, 10, 25)$ and $n = 104$. Then the algorithm would find

   3 coins of size 25
   2 coins of size 10
   1 coin of size 5
   4 coins of size 1

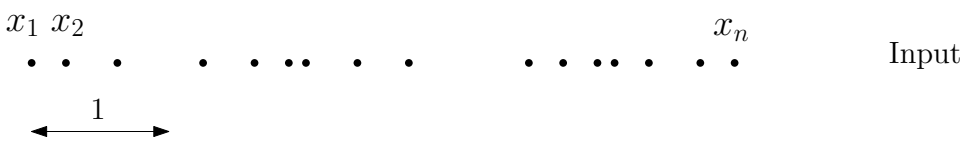
   (a) Prove that the greedy algorithm for change making always uses the smallest number of possible coins if the coin set consists of quarters (25 cents) , dimes (10) , nickels (5), and pennies (1).

   (b) Suppose that the available coins are in denominations that are powers of c. i.e. the denominations are $c^0, c^1, ..., c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

   (c) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. As noted, your set should include a penny so that there is a solution for every value of n.

7. **From CLRS:** A Greedy Algorithm.

   A *unit-length closed interval* on the real line is an interval $[x, 1 + x]$. Describe an $O(n)$ algorithm that, given input set $X = \{x_1, x_2, \ldots, x_n\}$, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct. You should assume that $x_1 < x_2 < \cdots < x_n$.



   As an example the points above are given on a line and you are given the length of a 1-unit interval. Show how to place a minimum number of such intervals to cover the points