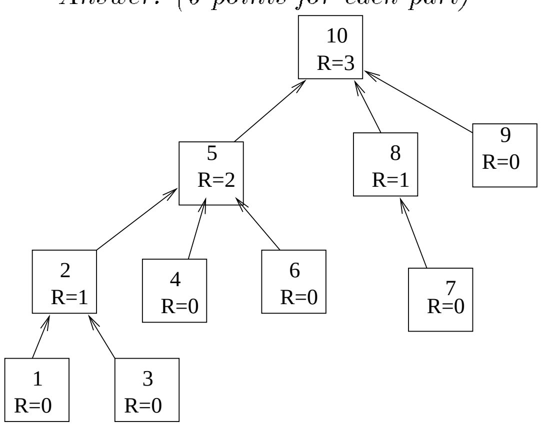
1. **(12 points)** The following describes an execution of MAKESET, UNION, and FIND operations on a set of 10 elements, labeled 1 through 10. MAKESET assigns rank 0 to an element, and UNION breaks ties by putting the tree whose root has the larger label as the parent of the other.

   for $j = 1$ to 10
       MAKESET(j)
   endfor
   UNION(1,2); UNION(1,3); UNION(4,5); UNION(4,6); UNION(1,6); UNION(7,8);
   UNION(9,10); UNION(7,10); UNION(3,8); FIND(4); FIND(3);

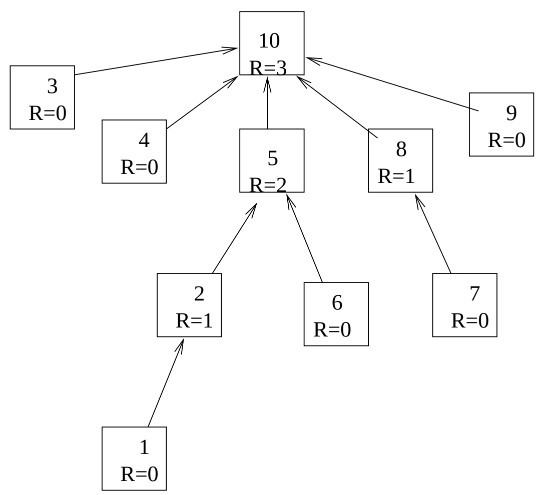
   - Give the tree from executing the above steps using union-by-rank *with no* path-compression. Be sure to label the nodes in the *final* tree, including their *final* ranks.
   - Give the tree from executing the above steps using union-by-rank *with* path-compression. Be sure to label the nodes in the *final* tree, including their *final* ranks.

   We recommend that you draw the intermediate trees for partial credit.

   Answer: (6 points for each part)



Without path compression



With path compression

2. **(16 points)** In this question we will consider how much Huffman coding can compress a file F of m characters taken from an alphabet of $n = 2^s$ characters $x_0, x_2, \ldots, x_{n-1}$.

- How many bits does it take to store F without using Huffman coding?

  Answer: (2 points) ms bits.

- Suppose $m = 1000$ and $n = 8$, with characters 0,1,2,3,4,5,6, and 7. Give an example of a file F (a string of 1000 digits from 0 through 7) in which every character x_i appears at least once, which compresses *the most* under Huffman coding. How many bits does it take to store the compressed file?

  Answer: (5 points) $F = 0123456777\cdots7$, i.e. 0 through 6 followed by 993 7s. 7 is encoded by 0, 0 by 100, and 1 through 6 by 1010 through 1111. The compressed file requires $6*4+1*3+993*1 = 1020$ bits, instead of 3000 bits uncompressed.

- Let $f(x_i)$ denote the frequency of x_i, i.e. the number of times x_i appears in F. Prove that there exist frequencies $f(x_i) > 0$ such that the number of bits needed to store F without Huffman coding is $\Omega(\log n)$ times the number of bits to store F when it is Huffman encoded. You can assume that the length of the file m, is much larger than n. Be sure to exhibit the bit patterns representing each character, both with and without Huffman coding, as well as explicit formulas for each $f(x_i)$.

  Answer: (9 points) As in the last part, one character appears $f(x_{n-1}) = m - n + 1$ times, and each other character appears $f(x_i) = 1$ time. x_{n-1} is encoded by 0, x_0 by $10\cdots0$ (1 followed by $s-1$ 0s), and x_1 through x_{n-2} by 1 followed by the usual s-bit encodings of 2 through $n-1$. The compressed file takes $(m-n+1)+s+(s+1)(n-2)$ bits to store. In contrast, in the uncompressed file each x_i is encoded by the usual s-bit pattern for i, and so it takes ms bits to store. The compression ratio is $(ms)/(m-n+1+s+(s+1)(n-2))$. When n is fixed and m is large, this ratio approaches $s = \log_2 n$ as desired.

3. **(20 points)** In class we derived the FFT for vectors of length n a power of two. In this question we will derive the FFT for $n = 3^s$, a power of three.

- Let $p(z) = \sum_{j=0}^{n-1} p_j \cdot z^j$ be a polynomial of degree at most $n-1$, where $n = 3^s$. Show that $p(z)$ can be written as the sum

$$p(z) = p0(z^3) + z \cdot p1(z^3) + z^2 \cdot p2(z^3) \tag{1}$$

  where $p0(z')$, $p1(z')$ and $p2(z')$ are each polynomials of degree at most $(n/3) - 1$. Be sure to explicitly exhibit the coefficients of each polynomial.

  Answer (5 points):

$$\begin{aligned} p0(z') &= p_0 + p_3 \cdot z' + p_6 \cdot z'^2 + \cdots + p_{n-3} \cdot z'^{n/3-1} = \sum_{j=0}^{n/3-1} p_{3j} \cdot z'^j \\ p1(z') &= p_1 + p_4 \cdot z' + p_7 \cdot z'^2 + \cdots + p_{n-2} \cdot z'^{n/3-1} = \sum_{j=0}^{n/3-1} p_{3j+1} \cdot z'^j \\ p2(z') &= p_2 + p_5 \cdot z' + p_8 \cdot z'^2 + \cdots + p_{n-1} \cdot z'^{n/3-1} = \sum_{j=0}^{n/3-1} p_{3j+2} \cdot z'^j \end{aligned}$$

- Let $\omega = e^{2\pi i/n}$, $i = \sqrt{-1}$, be a primitive n-th root of unity. Using equation (1), show that you can evaluate $p(z)$ at the n points ω^0, ω^1, ω^2, ... ,ω^{n-1}, *given* the values of the 3 polynomials $p0(z')$, $p1(z')$ and $p2(z')$ at the $n/3$ points ω^0, ω^3, ω^6, ω^9, ... , ω^{n-3}. You should write down a loop that evaluates $p'_j = p(\omega^j)$, for $j = 0$ to $n-1$, in terms of the values of $p0(z')$, $p1(z')$ and $p2(z')$.

  Answer (5 points):

  for $j = 0$ *to* $n/3 - 1$
  - $p'_j = p0(\omega^{3j}) + \omega^j \cdot p1(\omega^{3j}) + \omega^{2j} \cdot p2(\omega^{3j})$
  - $p'_{j+(n/3)} = p0(\omega^{3j}) + \omega^{j+(n/3)} \cdot p1(\omega^{3j}) + \omega^{2j+2(n/3)} \cdot p2(\omega^{3j})$
  - $p'_{j+2(n/3)} = p0(\omega^{3j}) + \omega^{j+2(n/3)} \cdot p1(\omega^{3j}) + \omega^{2j+4(n/3)} \cdot p2(\omega^{3j})$

  endfor

- Write a recursive subroutine for evaluating $p(z)$ at ω^j, $j = 0, ..., n-1$. Use your answer from the previous part in your answer.

  Answer (5 points):

  function FFTnp3(p) ... FFT for n a power of 3
  - n = *length(p)*
  - *if* $n = 1$ *return* p
  - $p0 =$ *FFTnp3*$((p_0, p_3, p_6, ..., p_{n-3}))$
  - $p1 =$ *FFTnp3*$((p_1, p_4, p_7, ..., p_{n-2}))$
  - $p2 =$ *FFTnp3*$((p_2, p_5, p_8, ..., p_{n-1}))$
  - $\omega = e^{2\pi\sqrt{-1}/n}$
  - *... insert loop from previous part*

- What is the complexity of your recursive subroutine? You should write down a recurrence for the complexity $T(n)$, justify it, and quote a theorem from class to solve it.

  Answers (5 points): $T(n) = 3T(n/3) + \Theta(n)$ *because each of the 3 recursive calls to FFTnp3 costs* $T(n/3)$*, and the loop over* j *costs* $\Theta(n)$*. By the general theorem about solving recurrences in class (with* $a = b = 3$, $c = 1$*), we get that* $T(n) = \Theta(n \log n)$.

4. **(18 points)** Give a divide and conquer algorithm for the following problem: you are given two sorted lists of size m and n and are allowed unit time to access the j-th element of each list. Give an $O(\log m + \log n)$ time algorithm for computing the kth largest element in the union of the two lists.
Give a recurrence relation for this problem and determine its complexity. Make sure you justify your recurrence relation and show your work when solving it. Hint: binary search.

Answer: Let $x_1, \ldots, x_m$, $y_1, \ldots, y_n$ be the two lists, sorted in decreasing order. Let $a = x_{m/2}$ and $b = y_{n/2}$. Further, let $a \leq b$. Then the number of elements $\leq b$ is at least $n/2 + m/2$. Further, the number of elements $\geq a$ is at least $n/2 + m/2$. Now, if $k \geq (n+m)/2$, then we can remove b and all elements bigger than it from the list of y_i's, and the solution would be the $(k - n/2)^{th}$ largest element in the remaining lists. Else, if $k < (m+n)/2$, then we remove a and all elements smaller than it from the list of x_i's, and find the k^{th} largest element in the remaining lists.

Since we are throwing away a constant fraction of the elements at each iteration, we have a running time of $O(\log m + \log n)$.

5. **(9 points)** No explanation required for these True/False questions, except for partial credit. Each correct answer is worth 1 point, but 1 point will be *subtracted* for each wrong answer, so answer only if you are reasonably certain.

- In a UNION-FIND data structure, a root node of rank three can have exactly one child.

  False. If it is a root node, the number of descendants will not be changed by any path compression.

- In UNION-FIND, the rank of a node can be equal to the rank of its parent.

  True. The parent of the root node is itself.

- In UNION-FIND, FIND with path compression can take a maximum of $\log(n)$ steps, where n is the number of elements.

  True.

- The algorithm for computing a Huffman code is an example of a greedy algorithm.

  True.

- The solution of $T(n) = 9T(n/2) + n^3$ is $\Theta(n^8)$.

  False. By the Master Theorem the answer is $\Theta(n^{\log_2 9}) = O(n^4)$.

- The solution of $T(n) = T(n-1) + n^4$ is $O(n^6)$.

  True. $T(n) = \frac{1}{5}n^5 + O(n^4)$*, which is also* $O(n^6)$.

- The solution of $T(n) = T(n-1000) + n^2$ is $O(n^3)$.

  True.

- The product $\omega^1 \cdot \omega^2 \cdot \omega^3 \cdots \omega^n$ of the n-th roots of unity is either 1 or -1 for all n.

  True.

- The coefficients of the polynomial $p(x) = \sum_{j=0}^{n-1} p_j \cdot x^j$ of degree at most $n-1$ are uniquely determined by the values $p(x_k)$ of the polynomial at the n points $x_0, \ldots, x_{n-1}$.

  False. All n *points must be distinct.*