1. (10 points) What are the minimum and maximum numbers of elements in a heap of height \( h \)? Justify your answer.

(From homework: Exercise 7.1-1, p 142 (6.1-1, p 129).)

Since a heap of height \( h \) is a binary tree of height \( h \) with all but the last level completely full, the minimum number of elements in a heap of height \( h \) is the number of elements in a complete binary tree of height \( h - 1 \) plus one extra element, or \( (2^h - 1) + 1 = 2^h \). The maximum number of elements in a heap is the maximum number of elements in a complete binary tree of height \( h \), or \( 2^{h+1} - 1 \).

2. (10 points) Given a hash table \( T \) with \( m \) slots that stores \( n \) elements from the universe \( U \). Show that if \( |U| > nm \), there is a subset of size \( n \) consisting of keys that all hash to the same slot, so that the worst-case searching time for hashing with chaining is \( \Theta(n) \).

(From homework: Exercise 12.2-6, p 226 (11.2-5, p 229).)

If \( |U| > nm \), then, by the Pigeonhole Principle, one slot must have more than \( \frac{nm}{m} = n \) keys. So, searching that slot using linear search (since there’s no guarantee that the list of keys is ordered) takes \( \Theta(n) \). Thus, the worst-case searching time for hashing with chaining is \( \Theta(n) \).

3. (15 points) Let \( \mathcal{F}(z) = \sum_{i=0}^{\infty} F_i z^i \) be the generating function for the Fibonacci recurrence, where \( F_i \) is the \( i \)th Fibonacci number \( (F_0 = 0, F_1 = 1, \text{ and } F_{i+2} = F_i + F_{i+1} \text{ for } i > 0) \).

(From Spring 1997 exam and from homework: Problem 4-6, p 74 (4-5, p 86).)

(a) Show that \( \mathcal{F}(z) = z + z\mathcal{F}(z) + z^2\mathcal{F}(z) \).

Using the definition of \( F_{i+2} \):

\[
F_{i+2} = F_i + F_{i+1} \\
F_{i+2}z^{i+2} = F_i + F_{i+1}z^{i+2} + F_i z^{i+2} \\
\sum_{i=0}^{\infty} F_{i+2}z^{i+2} = \sum_{i=0}^{\infty} F_i + \sum_{i=0}^{\infty} F_{i+1}z^{i+2} + \sum_{i=0}^{\infty} F_i z^{i+2} \\
= \sum_{j=1}^{\infty} F_j z^j + \sum_{i=0}^{\infty} F_i z^{i+1} + z^2 \sum_{i=0}^{\infty} F_i z^i \\
= \sum_{j=1}^{\infty} F_j z^j + z \sum_{j=1}^{\infty} F_j z^j + z^2 \mathcal{F}(z) \\
\mathcal{F}(z) - (F_0 + F_1 z) = z \sum_{j=1}^{\infty} F_j z^j + z^2 \mathcal{F}(z) \\
\mathcal{F}(z) - (0 + z) = z(\mathcal{F}(z) - F_0) + z^2 \mathcal{F}(z) \\
= z + z\mathcal{F}(z) + z^2 \mathcal{F}(z)
\]
Or, you can show this, by starting from the right hand side:

\[ z + zF(z) + z^2F(z) = z + z \sum_{i=0}^{\infty} F_i z^i + z^2 \sum_{i=0}^{\infty} F_i z^i \]
\[ = z + \sum_{i=0}^{\infty} F_i z^{i+1} + \sum_{i=0}^{\infty} F_i z^{i+2} \]
\[ = z + \sum_{j=1}^{\infty} F_{j-1} z^j + \sum_{j=2}^{\infty} F_{j-2} z^j \]
\[ = z + \sum_{j=1}^{\infty} (F_{j-1} + F_{j-2}) z^j \]
\[ = z + \sum_{j=1}^{\infty} F_j z^j \]
\[ = \sum_{j=0}^{\infty} F_j z^j \]
\[ = F(z) \]

Thus, \( F(z) = z + zF(z) + z^2F(z) \).

(b) Show that

\[ F(z) = \frac{z}{1-z-z^2} = \frac{1}{\sqrt{5}} \left( \frac{1}{1 - \phi z} - \frac{1}{1 - \hat{\phi} z} \right) \]

where \( \phi = \frac{1+\sqrt{5}}{2} \) and \( \hat{\phi} = \frac{1-\sqrt{5}}{2} \).

Solving \( F(z) = z + zF(z) + z^2F(z) \) for \( F(z) \) yields:

\[ F(z) - zF(z) - z^2F(z) = z \]
\[ F(z)(1-z-z^2) = z \]
\[ F(z) = \frac{z}{1-z-z^2} \]

Working backwards:

\[ \frac{1}{\sqrt{5}} \left( \frac{1}{1 - \phi z} - \frac{1}{1 - \hat{\phi} z} \right) = \frac{1}{\sqrt{5}} \left( \frac{1 - \phi z + \phi z^2}{1 - \phi z - \phi z^2} \right) = \frac{1}{\sqrt{5}} \left( \frac{1 - \phi z + \phi z^2}{(1 + \sqrt{5})/2} \right) = \frac{z}{\sqrt{5}} \left( \frac{1 - z(1 + \sqrt{5})/2 - z^2(1 + \sqrt{5})/2}{(1 + \sqrt{5})/2 - z^2(1 + \sqrt{5})/2} \right) = \frac{z}{\sqrt{5}} \left( \frac{1 - z(2/1 + \sqrt{5})/2 + z^2(1 + \sqrt{5})/2}{(1 + \sqrt{5})/2 - z^2(1 + \sqrt{5})/2} \right) = \frac{z}{\sqrt{5}} \left( \frac{1 - z/2 + z^2(1 - \sqrt{5})/4}{1/2} \right) = \frac{z}{2\sqrt{5}} \left( \frac{1 - z/2 + z^2(1 - \sqrt{5})/4}{1/2} \right) \]

(c) Show that \( F(x) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}} (\phi^i - \hat{\phi}^i) z^i \)

Show by induction on \( i \), or

using the identity: \( \forall c < 1, \sum_{i=0}^{\infty} c^i = \frac{1}{1-c} \), we have

\[ F(z) = \frac{1}{\sqrt{5}} \left( \frac{1}{1 - \phi z} - \frac{1}{1 - \hat{\phi} z} \right) = \frac{1}{\sqrt{5}} \left( \sum_{i=0}^{\infty} (\phi z)^i - \sum_{i=0}^{\infty} (\hat{\phi} z)^i \right) = \frac{1}{\sqrt{5}} \left( \sum_{i=0}^{\infty} \phi^i - \hat{\phi}^i \right) z^i = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}} (\phi^i - \hat{\phi}^i) z^i \]
4. **(20 points)** Justify each of your answers.  
*(From Spring 1998 exam, lecture notes, and the textbook.)*

(a) What is the best-case running time for quicksort?

Assume that the pivot chosen in the Partition() function divides the list exactly in half. Then, the running time would be:

\[
T(n) = 2 \cdot T(n/2) + \text{time to partition list}
\]

By the Master Theorem (or the substitution method), \( T(n) = O(n\lg n) \).
Since quicksort is a comparison-sort, the running time is bounded below by \( \Omega(n\lg n) \). So, \( O(n\lg n) \) is a tight bound and is the best-case running time for quicksort.

(b) What is the worst-case running time for quicksort?

If the pivot chosen only reduces the size of the lists by a constant amount, say 1, then the running time is:

\[
T(n) = T(1) + T(n - 1) + \text{time to partition list}
\]

\[
= T(1) + T(n - 1) + \Theta(n)
\]

\[
= T(1) + (T(1) + T(n - 2) + \Theta(n - 1)) + \Theta(n)
\]

\[
= \sum_{i=0}^{n} (T(1) + \Theta(n - i))
\]

\[
= O(n) + \sum_{j=0}^{n} \Theta(j)
\]

\[
= O(n) + O(n^2)
\]

\[
= O(n^2)
\]

5. **(20 points)** Given a list of integers \( A = [A_1, A_2, \ldots, A_n] \), define a zero-pair to be a sequence of two consecutive zeros in \( A \). Let \( NZ(A) \) be the number of zero-pairs in \( A \). For example, \( NZ([1, 4, 0, 0, 0, 2, 0, 0, 5]) = 3 \) (note that the 3 consecutive zeros in the list correspond to 2 zero pairs). Let \( NZ_k(A) = NZ([A_1, A_2, \ldots, A_k]) \) for \( 1 \leq k \leq n \).

*(From Spring 2000 exam.)*

(a) Write an algorithm to compute efficiently \( NZ_k(A) \).

```java
flag = false;
count = 0;
for ( i = 1; i <= n; i++ )
{   if ( A[i] == 1 )
    {   if ( flag )
        count++;
        flag = true; }
    else flag = false;
}
return(count);
```
(b) Analyze the running time of your algorithm.

```c
flag = false;
count = 0;
for ( i = 1; i <= n; i++)
{   if ( A[i] == 1 )
    {   if ( flag )
        count++;
        flag = true;
    }
    else flag = false;
}
return(count);
```

TOTAL TIME: \( O(n) \)

The list is traversed once, performing constant time operations each time, so the running time is \( O(n) \). This is the best possible, since you must look at every element at least once to tell if it is a 0 or not, which takes \( \Omega(n) \). Thus, the running time is \( \Theta(n) \).

6. (25 points) Let \( T \) be a binary search tree on \( n \) nodes. Define \( P(T) \) to be the internal path length of the tree \( T \) (that is, the sum, over all nodes \( x \) in \( T \) of the depth of \( x \), \( d(x,T) \)). We wish to show that the expected value of \( P(T) \) is \( O(n \lg n) \).

(From homework: Problem 13-3a-e, p 261 (12-3a-e, p 270).)

(a) Argue that the average depth of a node in \( T \) is

\[
\frac{1}{n} \sum_{x \in T} d(x,T) = \frac{1}{n} P(T)
\]

This follows from the definition, since the average depth of a node is the sum of all the depths, \( \sum_{x \in T} d(x,T) \) divided by the total number of nodes, \( n \).

(b) Let \( T_L \) and \( T_R \) denote the left and right subtrees of tree \( T \), respectively. Argue that if \( T \) has \( n \) nodes, then \( P(T) = P(T_L) + P(T_R) + n - 1 \).

Let \( x \in T_L \). Then, the path length of \( x \) in \( T \) is the path length of \( x \) in \( T_L \), plus one for the extra edge that connects the root of \( T_L \) to the root of \( T \). We have the similar result for all nodes in \( T_R \). Note that the height of the root of \( T \) (the only node not in \( P_L \cup P_R \)) is 0, and \( |P_L \cup P_R| = n - 1 \). This gives:

\[
P(T) = P(T_L) + P(T_R) + n - 1.
\]
(c) Let $P(n)$ denote the average internal path length of a randomly built binary search tree with $n$ nodes. Show that

$$P(n) = \frac{1}{n} \sum_{i=0}^{n-1} (P(i) + P(n - i - 1) + n - 1) = \frac{2}{n} \sum_{k=1}^{n-1} P(k) + \Theta(n).$$

Each binary search tree corresponds to a random permutation of the elements of the list $\{1, 2, \ldots, n\}$ (view the permutation as the order in which the elements are inserted into the tree). The probability of choosing $j$, $1 < j < n$, as the first element inserted into the tree is $\frac{1}{n}$. Note that if $i$ is chosen, the size of the left subtree is $j - 1$ and the size of the right subtree is $n - j - 1$. For $j = 1$, the size of left tree is 0 and the size of the right tree is $n - 1$. For $j = n$, the size of left tree is $n - 1$ and the size of the right tree is 0. Thus,

$$P(n) = \frac{1}{n} \sum_{i=0}^{n-1} (\text{first element is } i) = \frac{1}{n} \sum_{i=0}^{n-1} (P(i) + P(n - i - 1) + n - 1) = \frac{1}{n} (n - 1) + \sum_{i=0}^{n-1} (P(i) + P(n - i - 1)) = \frac{1}{n} ((n^2 - n) + 2 \sum_{i=0}^{n-1} P(i)) = 2 \sum_{k=0}^{n-1} P(k) + \Theta(n).$$

(d) Show that for $n > 2$:

$$\sum_{k=1}^{n-1} k \log k \leq \frac{1}{2} n^2 \log n - \frac{1}{8} n^2.$$

(This is shown in section on randomized quicksort in the book.) The easiest way to show this is by splitting the summation into 2 parts:

$$\sum_{k=1}^{n-1} k \log k = \sum_{k=1}^{[n/2]-1} k \log k + \sum_{k=[n/2]}^{n-1} k \log k.$$

The $\log k$ in the first summation on the right hand side is bounded above by $\log(n/2) = \log n - 1$. The $\log k$ in the second summation is bounded above by $\log n$. Thus, if $n \geq 2$,

$$\sum_{k=1}^{n-1} k \log k \leq (\log n - 1) \sum_{k=1}^{[n/2]-1} k + \log n \sum_{k=[n/2]}^{n-1} k \log k = \log n \sum_{k=1}^{n-1} k - \sum_{k=1}^{[n/2]-1} k \log k \leq \frac{1}{2} \log n - \frac{1}{2} (n - 1) \frac{n}{2} \leq \frac{1}{2} n^2 \log n - \frac{1}{8} n^2.$$

(e) Show that $P(n) = O(n \log n)$.

Show by induction that this holds for $n > 0$. It can easily be shown that the base case, $n = 1$, holds. So, we will focus on the inductive step: assume this is true for $k < n$ and show for
That is, we assume for $k < n$: $P(k) = O(k \lg k)$. This gives, summing over $k$, and by part d), that

$$
\sum_{k=1}^{n-1} P(k) = \sum_{k=1}^{n-1} O(k \lg k) \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2
$$

So,

$$
P(n) = \frac{1}{n} \sum_{k=0}^{n-1} P(k) + \Theta(n)
\leq \frac{1}{n} \left( \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \Theta(n)
= \left( \frac{1}{2} n \lg n - \frac{1}{8} n \right) + \Theta(n)
= O(n \lg n) + \Theta(n)
= O(n \lg n)
$$