1. (20) Consider the following program which build a heap in an array $A[1..n]$.

   **for** $(i = n/2;\ i \geq 1;\ i = i - 1)$ `sift`$(A,\ i)$;

   Note that we can also consider a heap to be a tree. Each tree rooted at $A[i]$ has its left sub-tree rooted at $A[2i]$ and right sub-tree rooted at $A[2i+1]$. In the above program, `sift`$(A,\ i)$ is a subprogram which will make the sub-tree rooted at $A[i]$ be a heap, provided that its two sub-trees are already heaps. The time needed by `sift`$(A,\ i)$ is known to be $O(h)$, where $h$ is the height of the sub-tree rooted at $A[i]$. Show that the time to make an array $A[1..n]$ into a heap by using the above method is $O(n)$.

2. (20) Quick-sort is a sorting algorithm which can be described as follows. It chooses a value $x$ and partitions the array to be sorted into two parts. The first part contains elements less than or equal to $x$ and the second part contains elements greater than or equal to $x$. It then sorts both parts recursively. A method to implement quick-sort non-recursively is to use a stack to store the ranges of the array yet to be sorted. The algorithm repeatedly sorts the part of the array whose range is stored on the top of the stack until the stack is empty. Show that if the algorithm stores the smaller part on top of the stack, then the height of the stack is bounded by $\log_2 n + 1$.

3. (20) You are given an infinite array $A[\cdot]$ in which the first $n$ cells contain integers in sorted order and the rest of the cells are filled with $\infty$. You are not given the value of $n$. Describe an algorithm that takes an integer $x$ as input and finds a position in the array containing $x$, if such a position exists, in $O(\log n)$ time. (If you are disturbed by the fact that the array $A$ has infinite length, assume instead that it is of length $n$, but that you don't know this length, and that the implementation of the array data type in your programming language returns the error message $\infty$ whenever elements $A[i]$ with $i > n$ are accessed.)

4. (40) In the proof of the theorem "Any algorithm for finding the median of $n$ elements $X = \{x_1, x_2, \ldots, x_n\}$ by comparison must do at least $3(n-1)/2$ comparisons." We construct a graph $G = (X, E)$, where $E$ represents the results of comparison of two elements in $X$. That is, if $x \leq y$, then there is an edge from $x$ to $y$.

   (a) (10) State the condition that the algorithm can determine the median by using the graph $G$.

   (b) (10) Define *critical* and *non-critical* edges.

   (c) (15) Assume that $n = 8$, that is, $X = \{x_1, x_2, \ldots, x_8\}$. Assume that the adversary assigns integers $1, 2, 3, 4, 5, 6, 7, 8$ to $x_i$, and initially the median is set to $x_4 = 4$. Suppose that the algorithm compares the elements shown in the following table.

   | elements compare | response | values | median | critical |
   |---|---|---|---|---|
   | - | - | $x_4 = 4$ | $x_4$ | - |
   | $x_1 : x_2$ | ? | $x_1 = ?,\ x_2 = ?$ | ? | ? |
   | $x_3 : x_4$ | ? | $x_3 = ?,\ x_4 = ?$ | ? | ? |
   | $x_1 : x_3$ | ? | $x_1 = ?,\ x_3 = ?$ | ? | ? |
   | $x_5 : x_6$ | ? | $x_5 = ?,\ x_6 = ?$ | ? | ? |
   | $x_7 : x_8$ | ? | $x_7 = ?,\ x_8 = ?$ | ? | ? |

   Your task is to fill out the above table to simulate what the adversary does. That is, in the *response* column, your answer must be the results of the comparison. For example, if you assign $x_1 = 1$ and $x_2 = 7$, your response must be $x_1 < x_2$. In the *values* column, your answer must be the least integer which can be assign to that valuable if this valuable is assigned a value less than the median. On the other hand, your answer must be the largest integer which can be assign to that valuable if this valuable is assigned a value greater than the median. In the *median* column, if a median needs to be changed, then write down the new median. Otherwise, write down a "-". In the *critical* column, if this comparison is critical, write down "Y". Otherwise, write down "N".

   (d) (5) At this time, can the algorithm determine the median of $X$ or not? Justify your answer.

5. (10) Design a linear time algorithm for testing whether an input graph is a bipartite graph or not.

6. (20) A tree is a connected undirected graph without cycles.

   (a) Show that a tree has at least two vertexes of degree 1.

   (b) Show that a tree of $n$ vertexes has $n - 1$ edges.