1. Recall the extension of Euclid's algorithm that we discussed in class.

```
1  def extended_Euclid(a,b):
2      """
3      a, b are non-negative integers
4      The function returns (u, v, d) such that d = gcd(a,b)
5      and d = ua + vb
6      """
7
8      if b == 0: return (1, 0, a)
9      (u, v, d) = extended_Euclid(b, a % b)
10     return (v, u - v * (a//b), d)
```

We argued in class that the algorithm correctly returns (u, v, d) as stated in the comment in the beginning of the code. Suppose for a certain input (a, b), where $a > b \geq 1$, the call to extended_Euclid(a, b) executes line 9 a total of t times (where $t \geq 1$). Let the value of (a, b) in the i-th call to extended_Euclid(a, b) be (a_i, b_i); let $(a_0, b_0) = (a, b)$. Let the value (u, v) returned by the i-th call be (u_i, v_i), so that $u_i a_i + v_i b_i = d$; thus $(u_t, v_t) = (1, 0)$. Then, for $i = 1, 2, \ldots, t$, we have

$$\begin{bmatrix} a_{i-1} \\ b_{i-1} \end{bmatrix} = \begin{bmatrix} q_i & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \end{bmatrix};$$

$$\begin{bmatrix} u_{i-1} & v_{i-1} \end{bmatrix} = \begin{bmatrix} u_i & v_i \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -q_i \end{bmatrix},$$

where q_i is the quotient obtained on dividing a_{i-1} by b_{i-1}.

(a) Show that $|u_i| \leq b_i/d$ and $|v_i| \leq a_i/d$, where $d = \mathsf{gcd}(a, b)$. You may use induction to show that the claim holds for $i - 1$ assuming it holds for i; what is the base case?

(b) Suppose a and b are n-bit integers. Show that the total number of bit operations needed for extended_Euclid(a, b) is $O(n^3)$, assuming that integer division of ℓ-bit integers can be done in using $O(\ell^2)$ bit operations.

2. Consider the following modification to Euclid's algorithm.

```
def modified_Euclid(a,b):
    """
    a, b are non-negative integers
    The function returns (u, v, d) such that d = gcd(a,b)
    and d = ua + vb
    """

    if b == 0: return a
    r = a % b
    if r < b/2:
        return modified_Euclid(b, r)
    else:
        return modified_Euclid(b, b-r)
```

(a) Argue that for integers $a > b > 0$, modified_Euclid(a, b) returns the gcd of a and b.

(b) How many times is modified_Euclid called recursively after modified_Euclid(a, b) is called with Fibonacci numbers $a = F_{t+1}$ and $b = F_t$?

3. Describe an algorithm to determine if a given positive number $N \geq 2$ can be written in the form $N = Q^E$, where Q and E are both integers at least 2. For n-bit numbers N, your algorithm should run in time $O(n^k)$ for some small constant k (fixed independent of n).

4. Suppose x, y and ℓ are n-bit numbers, such that $x > y$. Suppose the binary expansion of the fraction x/y is

$$0.b_0b_1b_2b_3\ldots = \sum_{i\geq 1} b_i 2^{-i},$$

which in general may not terminate. Describe an algorithm to determine b_ℓ, given x, y and ℓ. Your algorithm should run in time $O(n^k)$ for some small constant k (fixed independent of n).

(Due 30 Aug 2023)

5. Here is a problem closely related to quicksort, which we briefly discussed in class. Let X be a totally ordered set with at least n elements. Suppose $x_1, x_2, \ldots, x_n$ are drawn from X uniformly without replacement, and these elements are inserted into a *binary search tree* one after another.

   (a) There is an ordering for which $n(n-1)/2$ comparisons need to be made. How many such orderings are there?

   (b) Suppose $i > j$. We wish to determine the probability that x_i will be compared with x_j, when x_j is eventually inserted. Consider the distribution of $x_1, x_2, \ldots, x_i$ and x_j. Now, x_j is equally likely to appear in any of the $i+1$ gaps when $x_1, x_2, \ldots, x_i$ are arranged in sorted order. For x_i and x_j to be compared (when x_j is eventually inserted), in which gaps must x_j fall? (Notice how the answer to this part is related to the previous part.)

   (c) Conclude that the expected number of comparisons for building the binary search tree is precisely

$$\sum_{i=1}^{n} \left(\frac{2}{i+1}\right)(n-i),$$

   and show that this quantity is at most $2(n-1)\ln(n+1)$.

6. (a) $A[1..m]$ and $B[1..n]$ are two lists of integers sorted in ascending order. We wish to determine the k-th largest element in the union of A and B. Give an algorithm that runs in time $O(\log m + \log n)$. Assume that the $m+n$ elements are all distinct.

   (b) Suppose $A[1..m; 1..n]$ is an $m \times n$ array of integers. Suppose, first each row of A is sorted independently in ascending order from left to right; then, the columns of A are sorted independently in ascending order from top to bottom. Show that the rows of A remain sorted in the final array.

7. Problem 2.23 of [DPV].

8. Problem 2.32 of [DPV].

9. Suppose we are given a sequence $\mathbf{b} = b_0 b_1 \ldots b_{m-1} \in \{+1, -1\}^m$ called text and another shorter sequence $\mathbf{a} = a_0 a_1 \ldots a_{n-1} \in \{+1, -1\}^n$ called pattern (we use $\{+1, -1\}$ instead of $\{0, 1\}$), we say that $\mathbf{a}$ occurs in $\mathbf{b}$ at position j if $j \leq m - n$, and $a_k = b_{j+k}$ for $k = 0, 1, \ldots, n-1$. Notice that $\mathbf{a}$ occurs in $\mathbf{b}$ at position j iff $\sum_{k=0}^{n-1} a_k b_{k+j} = n$.
   (a) Describe polynomials $A(X)$ and $B(X)$ whose coefficients are derived from $\mathbf{a}$ and $\mathbf{b}$ such that by examining the coefficients of $C(X) = A(X)B(X)$, we can determine if $\mathbf{a}$ occurs in $\mathbf{b}$ at position j.
   (b) Now, suppose some of the elements of the pattern $\mathbf{a}$ are allowed to be $\star$, and we say that $\mathbf{a}$ occurs in $\mathbf{b}$ at position j if $j \leq m - n$ and ($a_k = \star$ or $a_k = b_{j+k}$) for $k = 0, 1, \ldots, n-1$. In this new setting, how would you modify the polynomials above to determine if $\mathbf{a}$ occurs in $\mathbf{b}$ at position j?
   (c) Based on the above, what method would you use to determine all positions j such that $\mathbf{a}$ occurs in $\mathbf{b}$ at position j. How long would it take? When is this method preferable to brute force search?

(Due 11 Sep 2023)

10. Suppose $G = (V, E)$ is an undirected unweighted graph with n vertices and m edges. Suppose $s, t \in V$ are vertices of G whose distance in G is strictly greater than $n/2$. Show that there is a vertex (other than s and t) whose deletion disconnects s from t. Describe an algorithm (assume that adjacency lists are available) running in time $O(m + n)$.

11. Suppose $G = (V, E)$ is a connected undirected graph. Suppose DFS starting at a vertex v and BFS starting at the same vertex v produce the same tree. Then, show that G is a tree.

12. Suppose G is a directed graph with n vertices and m edges. Describe an algorithm (assume adjacency lists are available) running in time $O(m + n)$ if G has a vertex v from where every other vertex is reachable.

13. Problem 3.28 (page 106) of [DPV].

14. Problem 4.19 (page 130) of [DPV].

(Due 27 Sep 2023)