6 Algorithms and Complexity

6.1 Fibonacci Numbers

1. **(Same as in Slides)**Is the number of the 100th Fibonacci number odd or even? Explain why.

2. **(Same as in Slides)** Make the programs fibr(k) and fibl(k). Confirm that they return the same values for some k's. Make a program that decides if fibr(k) and fibl(k) are same for k=1,...,p with a given p.

3. Let $\phi = \frac{1+\sqrt{5}}{2}$. It is known (see also Problem 5 below) that fib(k) is approximated by
$$\mathsf{fib(k)} \approx \frac{\phi^k}{\sqrt{5}}.$$
Define a function fiba(k) that computes the right-hand side of the equation, and examine the difference between fiba(k) and fibr(k) for k=1,...,p with a given p.

4. **(Same as in Slides)** Make the functions fibr(k) and fibl6(k), and record the computational times t_r and t_l for the two functions using run. Estimate A, B, C that satisfy $t_r(k) \simeq A \cdot B^k$ and $t_l(k) \simeq Ck$.

5. Explain why fibr(k) takes $4f(k) - 3$ operations. Hint: we can use either of the following strategies.
   - Show $T_r(k) = 4f(k) - 3$ if $k \geq 1$ by induction using $T_r(k) = T_r(k-1) + T_r(k-2) - 3$ and $T_r(0) = T_r(1) = 1$, or
   - Estimate the number of nodes and edges in the generated tree as in Slides.

6. Let f_k be the kth Fibonacci number. Then it holds that
$$\begin{pmatrix} f_{k+1} \\ f_k \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_k \\ f_{k-1} \end{pmatrix}.$$
   (a) Explain why the above equation holds.
   (b) Let A be the matrix in the form of
$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$
   Determine the eigenvalues λ_1 and λ_2 of A.
   (c) By the definition of eigenvalues, there exists a nonsingular(invertible) matrix U such that
$$A = U \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} U^{-1}$$
   Determine the matrix U and U^{-1}.

(d) By (a), we have

$$\begin{pmatrix} f_{k+1} \\ f_k \end{pmatrix} = A^k \begin{pmatrix} f_1 \\ f_0 \end{pmatrix} = A^k \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Using the equation in (c), express the kth Fibonacci number f_k with λ_1 and λ_2.

7. Define the function matpower(a, n) that computes the nth power of a matrix a. We may suppose that a is a 2×2 matrix and it is given as a 2-dimensional array.

(a) Define the function matmul(a,b) that computes the product of two matrices a and b.

Recall: When two matrices have size 2 by 2, it holds that

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} = \begin{pmatrix} a_{00}b_{00} + a_{01}b_{10} & a_{00}b_{01} + a_{01}b_{11} \\ a_{10}b_{00} + a_{11}b_{10} & a_{10}b_{01} + a_{11}b_{11} \end{pmatrix}.$$

(b) Define the function matsquare(a) that computes the square of a matrix a.

(c) Define the function matpower_loop(a,n) that computes the nth power of a matrix a using for-loop.

(d) We can observe that the power of matrices can be computed more efficiently. For example, for a matrix A, $A^{16} = (((A^2)^2)^2)^2$. Hence A^{16} can be obtained by taking a square four times. This reduces the number of matrix multiplications from 16 to 4. More precisely, we have the following.

$$A^n = \begin{cases} I & \text{if } n = 0, \\ (A^{n/2})^2 & \text{if } n \text{ is an even number with } n \geq 2, \\ A \times A^{n-1} & \text{if } n \text{ is an odd number,} \end{cases}$$

where I is the identity matrix (unit matrix). For example, if we have A^{20}, then we have

$$\begin{aligned} &A^{20} \Rightarrow (A^{10})^2 \Rightarrow ((A^5)^2)^2 \Rightarrow ((A \times A^4)^2)^2 \Rightarrow ((A \times (A^2)^2)^2)^2 \\ &\Rightarrow ((A \times ((A^1)^2)^2)^2)^2 \Rightarrow ((A \times ((A^1 \times A^0)^2)^2)^2)^2 \Rightarrow ((A \times ((A^1 \times I)^2)^2)^2)^2. \end{aligned}$$

Thus the number of matrix multiplications to obtain A^{20} is 6.

Using this relationship, make a recursive program matpower(a,n) that computes the nth power of a matrix a.

(e) Estimate the computational time complexity of matpower(a,n) using the order notation.

8. Define the function fibm(k) that computes the kth Fibonacci number using matpower(a,n) in the previous exercises. Furthermore, confirm that it returns the same values as fibl and fibr.

9. Define the function fibm6(k) that computes the first 6 digits of the kth Fibonacci number using matpower(a,n). Using bench.rb, confirm that the computational time is proportional to $\log k$, which means $O(\log k)$.

6.2 Computational Complexity and Sorting

1. **(Combination number, same in slides)** Estimate computational complexities of the two programs to compute the combination numbers in Exercises 5.1.11–12 using recursion and repetition, respectively.

2. A certain store has two software A and B to process experimental data. It is known that A can process in $O(N^2)$ time, while B can process in $O(N \log_2 N)$ time, when the data size is N. For 1000-record test data, Software A takes 1 second, while Software B takes 10 seconds. The target data has 1-million records. Which software is better to process the target data?

3. **(Simple Sort, same in slides)** Define a function min_index(a,i) that returns the index of the minimum value in a[i],...,a[n-1], where n is the length of a. Complete the program of the simple sort algorithm using this function.

4. **(Merge Sort, same in slides)** Make the missing parts in the program in Slides, and complete the function merge(a,b). Confirm that it works by executing merge([3,5,9], [1,4,6,7,8]) and merge([0,0.5,1.0], [0,0.9,1.0]).

5. Consider to compare the computational times of simple sort and merge sort. In the merge sort algorithm, we need to do some complicated tasks such as "making a new array." Hence it is perhaps expected that the merge sort algorithm has more time than the simple sort algorithm.

   (a) Make a randomly-generated sequence. This can be done by executing randoms(id, size, max) in a downloadable program randoms.rb.

   Remark: randoms(id, size, max) returns an array with size size, where each entry is a random value which is at least 0 and less than max. Note that if max is 1, it returns a real, and, if max is a positive integer larger than 1, it returns an integer. The integer(parameter) id means an index, and if id, max, size are all same, it returns the same array.

   (b) Use run(f, x, y) in bench.rb in a similar way to the Fibonacci case to measure the computational times.

   (c) Define the function compare_sort(max, step) to compare two sorting algorithms. This function first makes random arrays whose sizes are step, 2step, 3step, ..., max, and measure computational times for the simple sort and the merge sort.

```
load("./randoms.rb") # randoms(id, size, max)
load("./bench.rb") # run(function_na,e, r, v)
load("./simplesort.rb") # simplesort(a)
load("./mergesort.rb") # mergesort(a)

def compare_sort(max, step)
   for i in 1..(max/step)
      x = i*step
      a = random(i,x,1)
      run("simplesort", x, a)
```

```
            a = random(i,x,1)
            run("mergesort", x, a)
        end
    end
```

Using the function compare_sort(max,step), discuss the differences between two algorithms

6. **(Recursive Definition of Merge Sort)** By the following instruction, define the function mergesort_r(a) that executes the merge sort algorithm recursively.

   Let us first consider the final step of the merge sort algorithm. At this step, we have two sorted sequences p and q, where p is obtained by sorting the first half of a, and q is obtained from the latter half of a. We can use the function merge, which has already defined, to merge the two sorted sequences p and q.

   To obtain p and q, it suffices to do the merge sort recursively. That is, we apply the merge sort to a part of the array a. Thus we introduce the function doing "sort the 1st to rth elements in a".

   Let us consider defining the function merge_rec(a, l, r) that sorts a[l],...,a[r].

   - If l=r, then it returns an array with size 1 consisting of only a[l]. (We need not to sort)
   - If l<r, then we divide the array a into two parts, apply recursively the merge sort, and merge them. That is,
     (a) Define m=(l+r)/2.
     (b) Apply merge_rec to the elements from lth to mth and the elements from (m+1)th to rth, respectively. Denote the obtained sorted sequences by b and c.
     (c) Return the sequence obtained by executing merge to b and c.

   When we can define merge_rec(a, l, r), mergesort_r(a) can be defined by only calling merge_rec(a,0,a.length()-1).

6.3 Problems from Past Exams

1. **(Past Exam 2010)** Suppose that an array a has size n and contains m kinds of positive integers. We want to store all the distinct integers of a to another array b of size m, and also return the frequencies of occurrence in an array c of size m. For example, if a=[3,1,4,1,5,9,2,6,5,3], then n is 10 and m is 7. In this case, b contains [3,1,4,5,9,2,6], and c contains [2,2,1,2,1,1,1], which means that a has two 3's, two 1's, and so on.

   (a) The following program is a program to compute b and c from a. Describe the computational complexity using n and m. Note that the parameters b and c are supposed to be arrays of size m.We suppose that each entry in array b is initialized to be 0.

```
def intcount(a, b, c)
   for i in 0..(a.length()-1)
      x = a[i]
      j = 0
      while b[j] != 0 && b[j] != x
         j = j + 1
      end
      if b[j] == 0
         b[j] = x
         c[j] = 1
      else
         c[j] = c[j] + 1
      end
   end
end
```

   (b) Suppose that a is sorted, that is, elements in a are ordered in nondecreasing order. Modifying the above program, make a new function intcount(a,b,c) that runs in $O(n)$ time.

2. **(Past Exam 2011)** Let a be an array of N integers, denoted by a $= [x_0, x_1, \ldots, x_{N-1}]$. Let s(a, i, j) be the sum of the integers from a[i] to a[j-1]$(0 \leq i \leq j \leq N)$ (when $i = j$, define s(a, i, j)$= 0$). Answer the following questions.

   (a) When a = [8, -4, -5, 2, 4, -5, 5, 3, -7, 8], we have s(a, 0, 0)$= 0$ and s(a, 0, 1)$= 8$. Calculate s(a, 0, 2), s(a, 0, 3), and s(a, 0, 4).

   (b) Using N, describe the computational complexity (order) of an algorithm using simple iteration to compute s(a, 0, N).

   (c) Let mss(a, x, y) be the maximum value of s(a, i, j) for all i, j with x $\leq$ i $\leq$ j $\leq$ y (we suppose 0 $\leq$ x $\leq$ y $\leq$ N). We make a program mss(a, 0, N) as below by computing s(a, i, j) for all pairs i and j. Describe the computational complexity of mss(a, 0, N) using N.

```
def mss(a,x,y)
  m = 0
  for i in x..y
    for j in i..y
       m = max(m, s(a,i,j))
    end
  end
  m
end
```

(d) It is known that if mss(a, 0, z - 1) = s(a, x, z - 1) and s(a, x, z) < 0, then a[z-1] is not contained in the interval that gives mss(a, 0, y) (in other words, mss(a, 0, y) = s(a, i, j) then z does not satisfy i ≤ z ≤ j). Using this fact, we can make the following program mss0(a,m) to compute mss(a, 0, m).

```
def mss0(a,m)
  t = 0
  sum = 0
  for i in 0..(m-1)
    sum = sum + a[i]
    if sum > t
      t = sum
    else
      if sum < 0
        sum = 0
      end
    end
    # (*)
  end
  t
end
```

When we apply mss0(a, 10) with a=[8,-4,-5,2,4,-5,5,3,-7,8], calculate the values of sum and t at the end (∗) of each repetition using the following table.

i	0	1	2	3	4	5	6	7	8	9
sum										
t										

(e) Describe the complexity order of mss0(a,N) discussed in (d), using N.

3. **(Past Exam 2011)** Reading the Ruby program, answer the following.

```
def f(x,y)
  z = 1
  while y != 0
    while y % 2 == 0
      x = x * x
      y = y / 2
```

```
    end
    y = y - 1
    z = z * x
  end
  z
end
```

(a) Present the results when we call this function with parameters f(2,4) and f(3,5).

(b) Let c = f(a,b). Explain concisely the relationship between c and a, b. Note that a and b are supposed to be nonnegative integers.

4. **(Past Exam 2011)** Suppose that there is a point whose x-coordinate is nonnegative. The point moves in one step through linear distance 1 toward randomly-chosen direction. If the point bumps into the y-axis, it will be reflected completely. In this case, the moving distance is the sum of distances before and after the reflection, which has to be one. However, we have a hole from $(0, -1)$ to $(0, 1)$.

   Let (x, y) be the initial coordinates of the point. We describe a procedure escapesteps(x,y) to compute the number of steps until the point passes through the hole.

```
include(Math)

def escapesteps(x,y)
  n = 0
  escaped = false
  while !escaped
    r = rand()
    dx = cos(2*3.14159265358979*r)
    dy = sin(2*3.14159265358979*r)
    x1 = x + dx
    y1 = y + dy
    if x1 < 0
      if (A)
        escaped = true
      else
        (B)
        y = y1
      end
    else
      (C)
      y = y1
    end
    n = n + 1
  end
  n
end
```

Note that the function rand() returns a (pseudo)random real number between 0 and 1.

(a) (A) represents a condition that the point passes through the hole. Fill in (A).

(b) (B) and (C) update the x-coordinate. Fill in (B) and (C).

5. **(Past Exam 2012)**

(a) Explain what f(5) returns.

```
def f(n)
  if n >= 2
    n * f(n - 1)
  else
    1
  end
end
```

(b) The above function f is written using recursion. Without using recursion, redefine f using only "while" "if" "for".

(c) Explain what g(5) computes.

```
def g(n)
  if n >= 2
    g(n - 1) + g(n - 2)
  else
    1
  end
end
```

(d) The function h is the function that we rewrite the above function g without recursion. Fill in the blanks.

```
def h(n)
  result = make1d(n+1)
  i = (A)
  while (B)
    if i >= 2
      result[i] = (C)
    else
      result[i] = 1
    end
    i = (D)
  end
  result[ (E) ]
end
```

6. **(Past Exam 2012)** Suppose that an array a contains n integers (n $\geq$ 2). Consider computing the minimum of absolute values of differences between two entries in a. For example, if a = [9, 3, 5], the answer is 2.

   (a) Describe an algorithm whose complexity order is $O(n^2)$. You can answer using sentences or as a Ruby program.

   (b) Assume that elements in a are sorted in nondecreasing order. Describe an algorithm that runs in $O(n)$ time under the assumption.

   (c) Consider an algorithm whose complexity order is better than one in (a), not assuming that a is sorted. Explain the complexity order with some reason.

7. **(Past Exam 2013, PEAK)** Consider an algorithm to compute the greatest common divisor of two integers a and b (a<b). The greatest common divisor (gcd, for short) of a and b is defined to be the largest positive integer that divides the numbers a and b without a remainder. For example, the gcd of 12 and 18 is 6.

   (a) Since the greatest common divisor is between 1 and a, we can make the following function gcd_loop using repetition. Fill in the blanks $\boxed{\text{(i)}}$ to $\boxed{\text{(iii)}}$.

```
def gcd_loop(a,b)
   result = 0
   for i in 1..a
     if [(i)] && [(ii)]
         [(iii)]
     end
   end
   result
end
```

   (b) We denote by gcd(a,b) the greatest common divisor of a and b. Letting r be the remainder when b is divided by a, we have gcd(a,b)=gcd(r,a). Explain why this equation holds.

   (c) Based on the relationship described in (b), we can make the following recursive program gcd_r(a,b). Fill in the blanks. Note that each box may have multiple lines.

```
def gcd_r(a,b)
   if a == 0
    [(iv)]
   else
    [(v)]
   end
end
```

   (d) Suppose that a=273 and b=504. How many times do we call gcd_r in gcd_r(a,b)?

(e) We would like to discuss the computational complexity of the function `gcd_r(a,b)`. First, explain why we always have `b` $\geq$ `2r` for the remainder `r` of `b` when divided by `a`. Then describe how many times we need to call `gcd_r(a,b)` using `a` and `b`.