ASSIGNMENT 3: BRUTE FORCE AND RECURSIVE APPROACHES

due date: Sunday, 27 March 2022 (23.59 WITA)

Aturan pengerjaan tugas:

1. Kerjakan **semua** soal yang ada. Naskah ini terdiri dari 4 soal wajib dan 1 soal bonus.
2. Tugas boleh diketik/ditulis tangan (pastikan bisa dibaca), boleh menggunakan Bahasa Indonesia/Inggris. Hindari menggunakan tinta merah. Soal teori disimpan dalam format pdf. Jika menggunakan tulis tangan, harap discan (tidak difoto), kemudian dikompresi untuk memperkecil ukuran file.
3. Untuk tugas pemrograman (soal nomor 2 (a-g), 3 b, dan 4 d), buatlah dengan jupyter notebook. Harap mengurutkan program sesuai dengan urutan pada soal, dan berikan keterangan nomor soal pada setiap program. Penjelasan dapat ditulis dengan "markdown" yang tersedia pada jupyter. Untuk setiap program, tambahkan "**annotation**" atau "**comment**" pada bagian yang Anda rasa perlu, untuk dokumentasi/memudahkan pembacaan program Anda.
4. Saya menganjurkan Anda menulis program dalam python dengan jupyter notebook. Namun apabila Anda memilih menggunakan bahasa pemrograman lain (C, C++, atau java), mohon disampaikan ke saya. File pemrograman disimpan dalam 1 folder yang kemudian dikompres menjadi ekstensi zip. File terdiri dari (1) source code; (2) readme file yang menjelaskan bagaimana program Anda dapat dijalankan melalui terminal, serta rangkuman tentang apa yang Anda kerjakan dalam program tersebut; (3) file tambahan yang dibutuhkan untuk mengetes program Anda; (4) hasil eksperimen yang Anda kerjakan (format pdf atau png). Penamaan setiap file harus rapi.
5. Tugas teori dikumpulkan dalam format **pdf**, dan file program disimpan dalam 1 folder yang kemudian dikompres menjadi **zip**, dengan format penamaan tugas: **NamaLengkap_Kelas_NIM.extension**. Contoh: **GedeGanesha_6A_1610101001.pdf** atau **GedeGanesha_6A_1610101001.zip**. Pengumpulan tugas melalui e-learning Undiksha.
6. Anda diizinkan untuk berdiskusi dengan rekan Anda. Namun Anda harus menuliskan/menjelaskan jawaban Anda sendiri, dan paham dengan baik apa yang Anda tulis. Terutama untuk bagian pemrograman, Anda dilarang keras melakukan copy-paste program dari rekan Anda maupun dari internet! Anda siap bertanggung jawab terhadap hasil pekerjaan Anda. Hasil pekerjaan yang memiliki kemiripan yang tinggi dengan pekerjaan mahasiswa lain mempengaruhi poin penilaian.
7. Tugas dinilai berdasarkan kerapian penulisan tugas teori, kerapian dan kejelasan program komputer, dan kejelasan serta kesesuaian jawaban/penjelasan dengan pertanyaan yang diajukan. Total nilai maksimum tugas ini adalah 110. Keterlambatan dalam pengumpulan tugas mengurangi poin penilaian.

Dengan ini, Anda menyatakan bahwa Anda siap menerima segala konsekuensi jika nantinya ditemukan adanya kecurangan dalam pengerjaan tugas ini.

Problems

1. (*Brute force*, **15 poin**)

   Suatu perusahaan sedang melakukan uji coba untuk menentukan lantai tertinggi dari kantor pusatnya yang berlantai n, yang memungkinkan sebuah gadget dapat jatuh tanpa mengalami kerusakan. Perusahaan tersebut memiliki dua gadget identik untuk bereksperimen. Jika salah satunya rusak, maka gadget tersebut tidak dapat diperbaiki, dan eksperimen harus diselesaikan dengan gadget yang tersisa. (Catatan: kompleksitas waktu dihitung berdasarkan banyaknya percobaan yang dilakukan.)

   (a) (*5 poin*) Rancang sebuah algoritma dengan pendekatan brute-force untuk masalah ini. Jelaskan algoritmanya, kemudian tulis pseudocode untuk algoritma tersebut, dan tentukan kompleksitas waktunya!

   (b) (*10 poin*) Modifikasi algoritma tersebut untuk meningkatkan efisiensinya. (*Hint:* kita bisa membuat algoritma dengan kompleksitas waktu *worst-case* $\mathcal{O}(\sqrt{n})$.)

2. (*Have fun with recursion*, **20 poin**)

   Tulis program dalam bahasa python (buat di jupyter notebook) untuk latihan berikut (kerjakan soal secara berurutan). Pastikan untuk memperhatikan kasus dasar (*base case*) dan pemanggilan rekursi (*recursive call*) Anda!

   Tambahkan tabel evaluasi sebagai berikut di laporan file pdf Anda..

Nama program	**Poin 1**	**Poin 2**	**Poin 3**	**Poin 4**	**Keterbatasan program**
a. Integer multiplication					
b. Powering					
c. Print Down					
d. Print Up					
e. Reverse string					
f. Prime checking					
g. Fibonacci					

   *Poin penilaian pada tabel (diisi dengan **Ya/Tidak**)*

   1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)
   2. Program berhasil *running*
   3. Program dapat membaca file masukan dan menuliskan luaran.
   4. Program dapat mengatasi ketika input tidak sesuai dengan kriteria

   Spesifikasi program:

   (a) Misal a dan b adalah bilangan bulat tak negatif. Saat di SD, kita diajarkan bahwa nilai $b \times a$ ekuivalen $\underbrace{a + a + \cdots + a}_{\text{sebanyak } b}$. Manfaatkan sifat penjumlahan tersebut untuk membuat fungsi yang mengambil input dua bilangan bulat tak negatif dan mengalikannya secara rekursif.

   (b) Buat fungsi yang memberikan input bilangan bulat X dan $n \geq 0$, dan menghitung X^n secara rekursif. Anda tidak diperbolehkan menggunakan operator $**$ (operator pangkat pada python)!

   (c) Buat fungsi menggunakan rekursi untuk mencetak angka dari n ke 0.

(d) Modifikasi fungsi sebelumnya untuk membuat sebuah fungsi menggunakan rekursi untuk mencetak angka dari 0 hingga n.

(e) Tulis fungsi rekursif yang mengambil input sebuah string dan memberikan return string dalam urutan terbalik (Contoh: input = "Salam" maka output = "malaS'). Satu-satunya operasi string yang boleh Anda gunakan adalah penggabungan string (atau *concatenation*, dengan menggunakan operasi "+").

(f) Tulis sebuah fungsi rekursif untuk mengecek apakah suatu bilangan n adalah bilangan prima (Anda harus memeriksa apakah n habis dibagi dengan bilangan di bawah n).

(g) Tulis fungsi rekursif yang mengambil satu argumen $n \geq 1$ dan menghitung $F(n)$, yakni nilai ke-n dari barisan Fibonacci. Barisan Fibonacci didefinisikan oleh relasi:

$$F(n) = \begin{cases} 1, & n = 1 \\ 1, & n = 2 \\ F_{n-1} + F_{n-2}, & n \geq 3 \end{cases}$$

3. (*Permutation*, **30 poin**)

Permutasi adalah penyusunan kembali suatu kumpulan objek dalam urutan yang berbeda dari urutan yang semula. Misalnya, $A = [1, 2, 3]$ dapat dipermutasikan menjadi $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$, atau $[3, 2, 1]$ (6 macam permutasi). Diberikan algoritma permutasi sebagai berikut.

Algorithm 1 Unknown algorithm

```
1:  procedure UNKNOWN(A[0..n − 1], n)
2:      input: an array A = [1, 2, 3, . . . , n], and a positive integer n
3:      if n = 1 then
4:          print A
5:      else
6:          for i ← 0 to n − 1 do
7:              UNKNOWN(A[0..n − 1], n − 1)
8:              if n is odd then
9:                  swap A[0] and A[n − 1]
10:             else
11:                 swap A[i] and A[n − 1]
12:             end if
13:         end for
14:     end if
15: end procedure
```

(a) (*10 poin*) Jalankan algoritma tersebut dengan memberikan input $n = 2$ dan A adalah array dengan elemen integer $1, 2, 3, \ldots, n$. Tuliskan proses pada setiap iterasinya untuk setiap nilai n tersebut! Lakukan hal yang sama dengan nilai $n = 3$.

(b) (*5 poin*) Implementasikan algoritma tersebut dalam program komputer. Lalu tes dengan memberikan input integer n dan sebuah array $A = [1, 2, 3, \ldots, n]$. Tes program dengan nilai n yang berbeda. Jelaskan apa yang menjadi output program Anda (atau algoritma di atas)! Selidiki apakah hasil pada soal (a) sama dengan output yang diberikan program Anda!

Lengkapi tabel berikut, dan tulis dalam laporan di file pdf Anda. Tambahkan keterbatasan program setelah tabel.

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan (no syntax error)		
Program berhasil *running*		
Program dapat membaca file masukan dan menuliskan luaran		
Program memberikan output yang sama dengan soal (a)		
Program dapat mengatasi ketika input tidak sesuai dengan kriteria		

Keterbatasan program: _____
_____

(c) (*10 poin*) Berdasarkan jawaban pada soal (b), buktikan secara formal (dengan induksi matematika) kebenaran (*correctness*) algoritma tersebut!

(d) (*5 poin*) Tentukan kompleksitas waktunya (nyatakan dalam $\mathcal{O}(n)$). Jelaskan jawaban Anda!

4. (*Experiment with tower of Hanoi*, **35 poin**)

(a) (*5 poin*) Dalam versi asli "Tower of Hanoi problem", seperti yang diterbitkan pada tahun 1890-an oleh Édouard Lucas, seorang ahli matematika Prancis, dunia akan berakhir setelah 64 cakram dipindahkan dari Menara Brahma yang mistis. Berdasarkan algoritma yang didiskusikan di kelas, perkirakan berapa tahun yang diperlukan jika para bhikkhu dapat memindahkan satu cakram per menit. (Asumsikan bahwa para bhikkhu dapat bekerja sepanjang waktu, tidak makan, tidur, atau mati.)

(b) (*5 poin*) Berapa banyak gerakan yang dilakukan oleh piringan terbesar ke-i ($1 \leq i \leq n$) dalam algoritma ini? Jelaskan jawaban Anda!

(c) (*10 poin*) Algoritma rekursif "Tower of Hanoi":

- Implementasikan dalam bahasa python algoritma rekursif Anda. Masukan program adalah banyaknya cakram (atau n) dan list nomor cakram pada setiap tower, yaitu "source", "target", "helper". Output program adalah list elemen di ketiga tower "source", "target", "helper".
- Buat program rekursif "Tower of Hanoi", dengan spesifikasi: diberikan input n yaitu banyaknya cakram, program akan memberikan output berupa langkah-langkah memindahkan semua cakram dari "source" ke "target". Misalnya untuk $n = 2$, maka outputnya adalah:
  - Pindahkan cakram 1 dari source ke helper
  - Pindahkan cakram 2 dari source ke target
  - Pindahkan cakram 1 dari helper ke target

(d) (*15 poin*) Algoritma non-rekursif "Tower of Hanoi":

- Jelaskan secara algoritmik bagaimana Anda dapat menyelesaikan "Tower of Hanoi problem" dengan metode non-rekursif!
- Implementasikan dalam bahasa python algoritma non-rekursif Anda.
- Uji kedua program Anda dengan memberikan input banyaknya cakram $n = 3, 4, 5, 6, 7$ (atau, Anda dapat mengambil nilai n yang lebih besar jika Anda mau). Periksa apakah program yang Anda buat memberikan hasil yang sama dengan program rekursif sebelumnya. Jelaskan hasil pengamatan Anda!

- Buatlah sebuah eksperimen dengan memberikan input banyaknya cakram $n = 0, 1, 2, \ldots, 30$ (jika program Anda berjalan terlalu lambat, masukkan input hingga batas maksimum $n = 20, 15$ atau 10 disesuaikan dengan kecepatan program), untuk program rekursif Anda dan uji performance-nya dengan mencatatkan running time untuk setiap nilai n yang diinput.
- Buatlah eksperimen serupa untuk program non-rekursif.
- Visualisasikan running time masing-masing program dengan menyajikan grafik garis keduanya dalam satu bingkai, untuk membandingkan performance kedua program tersebut.
- Buat analisis dan simpulkan hasil eksperimen Anda!

Untuk memudahkan pengamatan, tambahkan tabel evaluasi sebagai berikut di laporan file pdf Anda. Tuliskan keterbatasan program di bawah tabel.

Poin	**Ya**	**Tidak**
Kedua program rekursif berhasil dikompilasi tanpa kesalahan (no syntax error)		
Kedua program rekursif berhasil *running*		
Kedua program rekursif memberikan output yang benar		
Program non-rekursif memberikan output yang benar		
Program rekursif pertama dan non-rekursif memberikan output yang sama jika diberikan input yang sama		
Program dapat menampilkan grafik perbandingan efisiensi		
Semua program dapat mengatasi ketika input tidak sesuai dengan kriteria		

Keterbatasan program: Jelaskan batas maksimum nilai n agar program Anda tidak *slow running*, dan hal-hal lainnya _____

5. (**Bonus:** *Mencari selebgram*, **10 poin**)

   Seorang selebgram di antara sekelompok n orang, adalah dia yang tidak mengenal siapa pun, tetapi dikenal oleh orang lain. Bagaimanakah cara mengidentifikasi orang yang merupakan selebgram dengan hanya mengajukan pertanyaan kepada orang-orang dalam bentuk "Apakah Anda mengenalnya?"

   1. Rancang sebuah algoritma yang efisien untuk mengidentifikasi selebgram atau menentukan bahwa suatu grup tidak memiliki orang seperti itu.
   2. Berapa banyak pertanyaan yang dibutuhkan algoritma Anda dalam kasus terburuk?