Homework 3

Vancouver Summer Program 2017
Algorithms and the Internet
Due: August 8, 2017, by 11:59 p.m.

---

1. (Simple graphs and simple properties.) Show that in a simple graph with at least two vertices, there must be two vertices that have the same degree.

2. (Dynamic corrections.) You are given an integer n and you want to determine if it is possible to transform n to a Fibonacci number (unless it is already a Fibonacci number) in at most m steps, where m is also a given bound. At each step, you can:

   - add 1 to the number at hand (addition step);
   - or multiply it by 2 (doubling step).

   As an example, if $n = 10$ and $m = 3$ then we know this is possible: in the first step, we multiply 10 by 2 and obtain 20. In the second step, we add 1 and obtain 21, which is a Fibonacci number.

   This problem can be solved using dynamic programming and here is one attempt to solve the problem:

```
// determine if n can be transformed to a Fibonacci number
// in at most m steps.
// At each step, we can add 1 or multiply by 2.
// Assume n > 0 and m >= 0
isPossible( n, m ) {
    if ( ( m is greater than or equal to 0 )
         and ( n is a Fibonacci number ) )
        return true
    else if ( m is 0 )
        return false
    return ( isPossible(n+2, m) || isPossible(2*n, m) )
}
```

   (a) Is the implementation shown above correct? If not, what change should you make to correct it?

   (b) If you were allowed to multiply by 2 no more than once (that is, you never multiply by 2 or do so exactly once), how would you change the implementation?

   (c) What is the asymptotic runtime complexity for checking if n is a Fibonacci number? Is this bounded by a polynomial in the size of the input?

3. (Identifying proctors.) The UBC Library permits students to reserve study spaces when they need to. The library also requires that there be student proctors to observe at least part of the usage of the study spaces. As an example, Sarah reserves a study space from 3pm to 5pm on Thursdays, Jo reserves a space from 4pm to 6pm on Thursdays, Martin reserves space from 10am-11am on Tuesdays, Sandeep reserves space from 10:30am to 12 noon on Tuesdays and Abdul reserves space from 11:30am to 1:00pm on Tuesdays. The librarian can then select Sarah, Martin and Sandeep as proctors because they can observe at least part of all the reservations. Proctors, once selected, are assumed to be disciplined enough to report on their own usage. Proctors may be paid for their efforts and therefore it is desirable to minimize the number of proctors. Given a list of names and reservations, describe an efficient algorithm to identify the smallest possible set of library proctors.

4. (Task allocation.) Suppose that there are n computational tasks $T_1, \ldots, T_n$ with exact memory consumption requirements $M_1, \ldots, M_n$ (in MB; not necessarily integers). To carry out the computational tasks, the owner of these tasks decided to choose a cloud service company called VspCloud. VspCloud decided to allocate three identical servers to handle the combined load of all the n programs. The servers are dedicated for the exclusive use of these programs. VspCloud would like to keep these servers as balanced as possible, so they want to distribute the computational tasks among the three servers such that the servers expend *exactly* the same amount of memory. As an algorithm designer at VspCloud, you are given the task of designing an algorithm to decide whether such task distribution is possible. Your algorithm should run in time that is polynomial in n and $\sum_{i=1}^{n} M_i$.

5. (What is happening here?) What does **Algorithm 1** below do? The input to the algorithm is an $n \times n$ 0-1 matrix. Analyze the running time of this algorithm and express your answer in Big-Oh. You may assume that matrix multiplication can be done in $O(n^{2.807})$ using Strassen's algorithm. You may want to see what the algorithm outputs on small examples. **Hint:** The algorithm runs in time $o(n^3)$; i.e, asymptotically *strictly* better than $O(n^3)$, so an answer of $O(n^3)$ as the running time will not get you full credit.

6. (True or False.) For each of the following statements, if the statement is true, briefly justify why, and if it is false, provide a counterexample.

   (a) There is a comparison-based sorting algorithm that on an array of n elements runs in $o(n \log n)$ time.

   (b) The number of arithmetic operations required to compute the $(n!)$th Fibonacci number is $\Theta(n \log n)$.

   (c) If a simple connected graph with $|V| \geq 2$ is triangle-free (no set of three vertices form a triangle), then it is not 4-colourable.

   (d) Every problem that is polynomial-time solvable is in the class **NP**.

   (e) If a decision problem is NP-Complete, then there is no polynomial-time algorithm that decides it.

Algorithm 1 $F(A)$

$Z \leftarrow A \cdot A$

Let B be an $n \times n$, 0-1 matrix, where

$$b_{i,j} = \begin{cases} 1 & \text{if } i \neq j \text{ and } (a_{i,j} = 1 \text{ or } z_{i,j} > 0) \\ 0 & \text{otherwise} \end{cases}$$

if $\forall i, j \; i \neq j$, $b_{i,j} = 1$ **then**
  return $D \leftarrow 2B - A$
end if
$T \leftarrow F(B)$
$X \leftarrow T \cdot A$
return $n \times n$ matrix D, where

$$d_{i,j} = \begin{cases} 2t_{i,j} & \text{if } x_{i,j} \geq t_{i,j} \cdot \sum_{k=1}^{n} a_{j,k} \\ 2t_{i,j} - 1 & \text{otherwise.} \end{cases}$$