THE UNIVERSITY OF BRITISH COLUMBIA
CPSC 320: MIDTERM #2 – Individual – November 7, 2018

Important notes about this examination

1. **You have 65 minutes to complete the 5 problems on this exam worth a total of 35 marks.**
2. You are allowed up to two sides of a letter-size sheet of notes as references. Otherwise no notes or aids are allowed. No electronic equipment is allowed.
3. Good luck!

Full Name:_____

Signature:_____

UBC Student #: |_|_|_|_|_|_|_|_|_|

Enter as your exam ID the rightmost 3 digits of the number printed in the corner of each page of your exam:

Exam ID: |_|_|_|

Student Conduct during Examinations

1. Each examination candidate must be prepared to produce, upon the request of the invigilator or examiner, his or her UBCcard for identification.
2. Examination candidates are not permitted to ask questions of the examiners or invigilators, except in cases of supposed errors or ambiguities in examination questions, illegible or missing material, or the like.
3. No examination candidate shall be permitted to enter the examination room after the expiration of one-half hour from the scheduled starting time, or to leave during the first half hour of the examination. Should the examination run forty-five (45) minutes or less, no examination candidate shall be permitted to enter the examination room once the examination has begun.
4. Examination candidates must conduct themselves honestly and in accordance with established rules for a given examination, which will be articulated by the examiner or invigilator prior to the examination commencing. Should dishonest behaviour be observed by the examiner(s) or invigilator(s), pleas of accident or forgetfulness shall not be received.
5. Examination candidates suspected of any of the following, or any other similar practices, may be immediately dismissed from the examination by the examiner/invigilator, and may be subject to disciplinary action:
    i. speaking or communicating with other examination candidates, unless otherwise authorized;
    ii. purposely exposing written papers to the view of other examination candidates or imaging devices;
    iii. purposely viewing the written papers of other examination candidates;
    iv. using or having visible at the place of writing any books, papers or other memory aid devices other than those authorized by the examiner(s); and,
    v. using or operating electronic devices including but not limited to telephones, calculators, computers, or similar devices other than those authorized by the examiner(s)— (electronic devices other than those authorized by the examiner(s) must be completely powered down if present at the place of writing).
6. Examination candidates must not destroy or damage any examination material, must hand in all examination papers, and must not take any examination material from the examination room without permission of the examiner or invigilator.
7. Notwithstanding the above, for any mode of examination that does not fall into the traditional, paper-based method, examination candidates shall adhere to any special rules for conduct as established and articulated by the examiner.
8. Examination candidates must follow any additional examination rules or directions communicated by the examiner(s) or invigilator(s).

CPSC 320 2018W1, Midterm 2

WRITE UGRAD IDs HERE (**-1 mark** if any missing or incorrect).

UGRAD ID:

1 Greed is good

1. **[1.5 marks]** Which one(s) of the following algorithms, listed here in alphabetical order, is/are greedy? Circle all that apply.
   - ◯ Dijkstra's shortest paths algorithm
   - ◯ Gale Shapley's stable matching algorithm
   - ◯ Kruskal's minimum spanning tree algorithm
   - ◯ The Power algorithm from quiz 4
   - ◯ The Unweighted Interval Scheduling algorithm
   - ◯ The Weighted Interval Scheduling algorithm

2. **[2 marks]** A server has n customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer i. So if, for instance, the customers are served in order of increasing i, then customer number i has to wait $\sum_{j=1}^{i} t_j$ minutes. We wish to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i).$$

   Describe succintly an efficient algorithm for computing the optimal order in which to process the customers.

3. **[2.5 marks]** The following is the outline of a proof of correctness for the correct solution to question 1.2. Please fill in the holes we left in the proof outline with the appropriate information.

   First, we prove that there is an optimal solution that

   [ ]

   Then we use

   [ ]

   and the previous fact to prove that our greedy solution is at

   [ ]

   solution.

2 Yet another spanning algorithm

Let $G = (V, E)$ denote a connected, undirected graph with $n \geq 2$ nodes and m weighted edges. Throughout this problem, *assume that all edges of E have distinct weights.* Let $\text{wt}(e)$ denote the weight of edge e of G. The following algorithm appeared on a recent quiz:

```
1.      Algorithm Spanning(G = (V, E), wt())
2.
3.          Let G' = (V, E') where E' = ∅
4.          While G' is not connected
5.              E-new = ∅
6.              For each connected component C of G' = (V, E')
7.                  Find an edge e = (u, v) ∈ E of minimum weight wt(e) that
8.                  connects a node u in C to a node v that is not in C
9.                  E-new = E-new ∪{e}
10.             E' = E' ∪ E-new
11.         Return G'
```

1. **[2 marks]** As covered in the quiz, in the worst case, $\Theta(\log n)$ iterations of the While loop will be executed. Describe an input graph with n nodes on which this worst case behaviour can happen, where furthermore the most costly edge of E is added to E' in the *first* iteration of the While loop.

2. **[4 marks]** Explain why the algorithm always returns a tree on all inputs $G = (V, E)$, given our assumption that all edges of E have different weights.

3 Recursive multiplication recurrence

[5 marks] The following algorithm, due to Karatsuba, multiplies two n-bit unsigned integers $x = x_n x_{n-1} \ldots x_1$ and $y = y_n y_{n-1} \ldots y_1$. For $n > 1$ the algorithm is based on the following observation. Let $x_H = x_n x_{n-1} \ldots x_{\lfloor n/2 \rfloor + 1}$ and $x_L = x_{\lfloor n/2 \rfloor} \ldots x_1$. For example, if $x = 11101$ then $x_H = 111$ and $x_L = 01$. Define y_H and y_L similarly, by breaking y in two. Then the product of x and y can be written as

$$\begin{aligned} xy &= x_H y_H 2^n + (x_H y_L + x_L y_H) 2^{n/2} + x_L y_L && (1) \\ &= z_2 2^n + z_1 2^{n/2} + z_0, && (2) \end{aligned}$$

where $z_0 = x_L y_L, z_1 = x_H y_L + x_L y_H$, and $z_2 = x_H y_H$. Furthermore, in equation (2), the multiplications by 2^n and $2^{n/2}$ can be done using bit-shift operations in $\Theta(n)$ time, and also the additions can be done in $\Theta(n)$ time.

```
Algorithm Multiply(x = x_1 x_2 ... x_n, y = y_1 y_2 ... y_n)
    // x and y are unsigned n-bit numbers, where n ≥ 1
    If n == 1 then // Base case
        If (x_1 == 1) and (x_2 == 1) then
            Return 1
        Else
            Return 0
    Else
        x_H = x_n x_{n-1} ... x_{⌊n/2⌋+1}
        x_L = x_{⌊n/2⌋} ... x_1
        y_H = y_n y_{n-1} ... y_{⌊n/2⌋+1}
        y_L = y_{⌊n/2⌋} ... y_1
        z_0 = Multiply(x_L, y_L)
        z_2 = Multiply(x_H, y_H)
        z_1 = Multiply(x_L + x_H, y_L + y_H) − z_0 − z_2 // This quantity is x_H y_L + x_L y_H
        Return z_2 2^n + z_1 2^{n/2} + z_0 // This step can be done in Θ(n) time
```

Let $M(n)$ be the running time of this algorithm on two n-bit integers. Give a recurrence that provides a good upper bound for $M(n)$. You can ignore floors and ceilings.

$$M(n) \leq \begin{cases} \boxed{} & \text{if } n = 1 \\ \boxed{} & \text{if } n \geq 2 \end{cases}$$

4 Recursive running times

[8 marks] While trying to come up with questions to ask on a midterm, Anne and Patrice discovered an amazingly novel algorithm to predict whether or not it will snow on the day of the final exam, based on n days worth of weather data. The time complexity of their algorithm is given by the recurrence relation

$$T(n) = \begin{cases} 2T(n/3) + T(2n/3) + cn^2 & \text{if } n \geq 2 \\ c & \text{if } n = 1. \end{cases}$$

Using a method of your choice, prove upper and lower bounds on the function $T(n)$. Your grade will depend partly on the quality of the bound you provide (so, proving that $T(n) \in \Omega(1)$ and $T(n) \in O(100^n)$, while true, will not give you many marks).

5 Pell numbers

Recall the Pell numbers, defined by the following recurrence relation, and the algorithm Pell(n) which requires exponential time to compute the nth Pell number:

$$P_n = \begin{cases} 0, & \text{if } n = 0, \\ 1, & \text{if } n = 1, \\ 2P_{n-1} + P_{n-2}, & \text{otherwise.} \end{cases}$$

Algorithm Pell(n)
// Returns the nth Pell number
    If $n = 0$ then
        Return 0
    ElseIf $n = 1$ then
        Return 1
    Else
        Return 2 * Pell($n - 1$) + Pell($n - 2$)

1. **[3 marks]** Use memoization to obtain a more efficient recursive algorithm for calculating Pell numbers.

```
Memo-Pell(n):  // n is nonnegative
  Create a new array Soln[0,1,... n] of length n+1

  Initialize Soln[0] to: _________

  If n >= 1 : Initialize Soln[1] to: ____________

  If n >= 2 : Initialize each element Soln[i] for 2 <= i <= n to -1
              // Here, -1 is a flag indicating that Soln[i] is not yet computed

  Return PellHelper(n, Soln)


PellHelper(n, Soln):

 If (Soln[n] == -1):   // Soln[n] is not yet computed
   // Recursively compute and store the answer
   // FILL IN THE CODE NEEDED HERE



 // By this point, Soln[n] is computed
 // FILL IN THE CODE NEEDED HERE
```

2. **[1 mark]** What is the running time of your Algorithm Memo-Pell from part 1? Check one.

   ◯ $\Theta(n)$ ◯ $\Theta(n \log n)$ ◯ $\Theta(n^2)$ ◯ $\Theta(2^n)$ ◯ $\Theta(3^n)$

3. **[3 marks]** Rewrite your Memo-Pell algorithm without using recursion.

4. **[1 mark]** What is the running time of your algorithm from part 3? Check one.

   ◯ $\Theta(n)$ ◯ $\Theta(n \log n)$ ◯ $\Theta(n^2)$ ◯ $\Theta(2^n)$ ◯ $\Theta(3^n)$

5. **[2 marks]** The previous algorithms use array Soln, which has $n + 1$ entries and thus uses $\Theta(n)$ memory. Briefly describe how to obtain an algorithm with the same running time that uses $O(1)$ memory.