CSCM12 – Software concepts and efficiency February 17th 2025
Cécilia Pradic

Coursework 1

1. Submission for this coursework will be individual; detailed submission instructions will be given on Canvas.

2. By submitting coursework, you state that you fully understand and are complying with the University's policy on Academic Integrity and Academic Misconduct. The policy can be found at `https://myuni.swansea.ac.uk/academic-life/academic-misconduct`. The consequences of committing academic misconduct can be extremely serious and may have a profound effect on your results and/or further progression. The penalties range from a written reprimand to cancellation of all of your marks and withdrawal from the University.

3. There is a total of 40 marks to be gained by answering the questions.

4. You may get up to 5 bonus marks bonus for signing off labs 1,2,3 and 4 (challenge tasks do not count). The total will still be over 40.

5. Please do this seriously and write things clearly – this is meant to be an opportunity to get ready for the exam and get some feedback on your individual written work before then. The other coursework will be purely coding.

Exercise 1 (6 marks)**.** 1. For each of the following assertation say whether they are true or false

   (a) $2n^2 + \log(n) = \mathcal{O}(n^3)$
   (b) $2n^2 + \log(n) = \mathcal{O}(n^2)$
   (c) $2n^2 + \log(n) = \mathcal{O}(\sqrt{n})$
   (d) $n^{200} = \mathcal{O}(2^n)$
   (e) $\log(n) = \mathcal{O}(1)$

2. Give a number K such that $\log(n)^{10} \leq Kn$ for every $n \geq 1$ (you do not need to provide a proof; you can help yourself of a computer algebra system to find the number K that you want).

Exercise 2 (14 marks)**.** For each of the following java function, give the asymptotic complexity in function of the input. Justify your answer.

1.
```java
static int bla1(int n) {
    int result = 0;
    for (i=0,i++,i<n) {
       result += i;
    }
    return result;
}
```

2.
```java
static int bla2(int n) {
    int result = 0;
    for (i=0,i++,i<n) {
```

```
            result -= bla1(i);
          }
          return result;
      }
```

3.
```
static int[] fun1(int n)
{
  if(n == 0)
    return new int[2];
  int[] bla = fun1(n-1);
  bla[0] += 1 + bla[1];
  bla[1] *= 3;
  return bla;
}
```

4.
```
    static int bla3(int n) {
       if (n <= 0) {
          return 0;
       }
       else {
          return bla3(n) + bla3(n-1);
       }
    }
```

5.
```
   static boolean bla4(int n) {
      if (n == 1) {
         return true;
      }
      else if (n % 2 == 1) {
         return false;
      }
      else {
         return bla4(n /2);
      }
   }
```

6.
```
static void fun3(int[][] arr)
{
  final int n = arr.length;
  if(n == 0 || arr[0].length != n)
    return;
  for(int k = 0; k < n; k += 2)
  {
    if(k%2 == 0)
      for(int j = n; j > 0; --j)
        arr[k][j] = arr[j][k];
    else
      for(int j = 0; j < Math.sqrt(n); ++j)
        arr[j*j][k] = arr[k-1][j];
  }
}
```

7.
```
static int fun4(int n)
{
  if(n <= 5)
    return d;
  int r = 0;
  for(int i = n; i >2; --i)
    r = (n + 8 * r) % 3;
  return (fun4(n/3) + fun4(n/3 - 1) * r) % 55;
}
```

Exercise 3 (11 marks)**.** Let's define the algorithmic change problem as follows:

- **Input:** A sequence of integers $c_0 = 1 < c_1 < \ldots < c_k$ representing *coin* values and a number a
- **Output:** An repartition of coins $r_0, \ldots, r_k$ such that giving giving back r_i coins of values c_i for all i yields the desired amount a (i.e. $\sum_i r_i c_i = a$)

1. Write pseudo-code for a greedy algorithm implementing the following idea: if we try to give back amount a, pick the largest i such that $c_i \leq a$; give back one coin of value c_i and proceed to produce the change of value $a - c_i$. (4 marks)
2. What is the complexity of that algorithm? (2 marks)
3. Call an answer to an answer *optimal* if it has the minimal amount of coins ($\sum_i r_i$) amongst all answers.

   If we use the coin system $c_0 = 1, c_1 = 4, c_2 = 5$, there is an amount a such that the greedy algorithm above does not return an optimal answer on the corresponding instance. Give an example of such an a and justify briefly your answer (2 marks, no need to write more than 3 sentences/formulas).
4. If $2c_i \leq c_{i+1}$ for all $i < k$, then the greedy algorithm always returns the optimal answer. Explain why. (1 mark)
5. Using dynamic programming, write pseudo-code or code for an algorithm that returns optimal solutions for all possible coin systems (for this, you may want to use an array of array of dimensions $a+1 \times k+1$). What is its complexity (dependent on the number n of coins and the amount of change a)? (3 marks)

Exercise 4 (9 marks)**.** Given an array A of, say integers, of length n (indexed from 0 to $n-1$), call *an inversion in* A a pair of integers (i, j) such that $0 \leq i < j < n$ and $A[i] > A[j]$. We are now interested in the complexity of counting the number of such inversions which we write $\mathrm{Inv}(A)$.

1. What is the number of inversions in the following three arrays? (2 marks)

$$[1, 3, 5, 8, 2, 4, 16, 20] \quad , \quad [1, 3, 5, 8] \quad \text{and} \quad [16, 2, 20, 4]$$

2. Write the pseudo-code for a naive algorithm that counts the number of inversions of an array of size n running in quadratic time ($\mathcal{O}(n^2)$). (2 marks)

3. Say that an array A is half-sorted when the first half and its second half are sorted. For instance, the array $[1, 3, 5, 8, 2, 4, 16, 20]$ is half-sorted because $[1, 3, 5, 8]$ and $[2, 4, 16, 20]$ are sorted.

   Write an algorithm that count the number of inversions in half-sorted arrays that runs in linear time ($\mathcal{O}(n)$). (2 marks)

4. For two arrays A and B, write $A^\frown B$ for their concatenation and $\mathrm{Sort}(A)$ for the sorted version of A. Then with a bit of reasoning, we can show that we always have:

$$\mathrm{Inv}(A^\frown B) \quad = \quad \mathrm{Inv}(A) + \mathrm{Inv}(B) + \mathrm{Inv}(\mathrm{Sort}(A)^\frown \mathrm{Sort}(B))$$

   Using a divide-and-conquer approach, the answer to the previous question[1] and the above formula, deduce a recursive algorithm that take as input an array A and outputs both the number of inversions in A and a sorted version of A. (2 marks)

5. What is the complexity of your divide-and-conquer algorithm? Why? (1 mark)

[1] You may simply assume the existence of an algorithm counting the number of inversions in linear time if you did not manage to answer the question.