CPSC 320 2023S2: Quiz 4 Solutions

1 Pell numbers (11 points)

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:

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

function Pell(n)
  ▷ Returns the nth Pell number
  if $n == 0$ **then**
    return 0
  else if $n == 1$ **then**
    return 1
  else
    return $2 \times$ Pell($n-1$) + Pell($n-2$)

1. (3 points) Complete the following memoization algorithm to obtain a more efficient recursive algorithm for calculating Pell numbers, by filling in the *two* missing initializations in Memo-Pell and the *two* missing pieces of code in Pell-Helper.

   SOLUTION:

   procedure Memo-Pell(n): ▷ n is nonnegative
     create a new array Soln$[0, 1, ...n]$ of length $n+1$
     Soln$[0] \leftarrow 0$
     Soln$[1] \leftarrow 1$
     if $n \geq 2$ **then**
       for $2 \leq i \leq n$ **do**
         Soln$[i] \leftarrow -1$
     return PellHelper(n)

   procedure Pell-Helper(n)
     if Soln$[n] == -1$ **then** ▷ Soln$[n]$ is not yet computed
       ▷ Recursively compute and store the answer
       Soln$[n] \leftarrow 2\times$ Pell-Helper($n-1$) + Pell-Helper($n-2$)
     ▷ By this point, Soln$[n]$ is computed
     return Soln$[n]$

2. (1 point) 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 points) Rewrite your Memo-Pell algorithm to get a dynamic programming algorithm that does not use recursion.

SOLUTION:

```
procedure DP-PELL(n): ▷ n is nonnegative
    create a new array Soln[0,1,... n] of length n + 1
    initialize Soln[0] to 0
    initialize Soln[1] to 1
    for i from 2 to n do
        Soln[i] ← 2 × Soln[i − 1] + Soln[i − 2]
    return Soln[n]
```

4. (1 point) 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. (3 points) The previous algorithms use array Soln, which has $n+1$ entries and uses $\Theta(n)$ memory. Complete the following pseudocode to obtain an algorithm with the same running time that uses $O(1)$ memory. The code uses two variables: CurrentValue and PrevValue to store the values of Soln[i] and Soln[$i-1$] in the ith iteration.

SOLUTION:

```
procedure MEMORY-EFFICIENT-PELL(n) ▷ n is nonnegative
    if n == 0 then
        return 0
    else
        initialize prevValue to 0
        initialize currentValue to 1
        for i from 2 to n do
            nextValue ← 2 × currentValue + prevValue
            prevValue ← currentValue
            currentValue ← nextValue
        return currentValue
```

2 k-means clustering (5 points)

Recall that the k-means clustering problem is to partition a non-empty set $S = \{x_1, x_2, \dots, x_n\}$ of data points into $k \le n$ non-empty clusters of nearby points. Here we assume that each point x_i is simply a real-valued number and that the points are ordered, with x_1 being the smallest and x_n being the largest. The points need not be distinct. More precisely, we want to find clusters that minimize the quantity

$$\sum_{1 \le i \le k} \sum_{x_j \in C_i} (x_j - \mu(C_i))^2,$$

where $\mu(C_i)$ denotes the mean of the set C_i of points. Moreover, each non-empty cluster C_i should be comprised of consecutive points from the list S.

1. (1 point) Let $F(i, l)$ be the cost of an **optimal** solution to the subproblem when the data points are $\{x_1, \dots, x_i\}$ and the number of clusters is l. Which of the following is the correct interpretation for $F(n-i, l)$? Choose one.

   $F(n-i, l)$ is the cost of an optimal solution to the subproblem when...

   - [x] ... the data points are $\{x_1, \dots, x_{n-i}\}$ and the number of clusters is l.
   - [ ] ... the data points are $\{x_{n-i}, \dots, x_n\}$ and the number of clusters is l.

2. (2 points) Which of these is correct expression for $F(n, k)$, when $1 \le k \le n$? Choose one.

   SOLUTION:

   - [x] $F(n, k) = \min_{0 \le i \le n-1}\{F(i, k-1) + \sum_{j=i+1}^{n}(x_j - \mu(\{x_{i+1}, \dots, x_n\}))^2\}$
   - [ ] $F(n, k) = \min_{0 \le i \le n-1}\{F(i, k) + \sum_{j=i+1}^{n}(x_j - \mu(\{x_{i+1}, \dots, x_n\}))^2\}$
   - [ ] $F(n, k) = \min_{0 \le i \le n-1}\{F(i, \lfloor k/2 \rfloor) + F(n-i, \lceil k/2 \rceil)\}$

3. (2 points) What are the base cases for $F(n, k)$? Make sure to consider all cases that could be needed by your chosen recurrence in the previous part.

   SOLUTION:

   $F(0, 0) = 0$

   $F(n, k) = \infty$, if $k > n \ge 0$ or $n > k = 0$

3 SAT to 3SAT (4 points)

Consider the Transform-Instance algorithm from the worksheet, for converting SAT instances to 3SAT instances. It is summarized below.

1. (1 point) When the SAT instance is

$$(x_1 \vee x_2 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_3 \vee x_4 \vee x_5 \vee x_6) \wedge (\bar{x}_6) \wedge (\bar{x}_2 \vee x_3),$$

how many clauses are in the resulting 3SAT instance?

● 6 ○ 7 ○ 8 ○ 9 ○ 10

2. (1 point) For the same example as part 1, how many distinct variables are in the resulting 3SAT instance? (Multiple copies of some variable x_i or its complement $\bar{x}_i$ count as one variable.)

○ 6 ○ 7 ● 8 ○ 9 ○ 10

3. (1 point) More generally, suppose that the reduction maps SAT instance I to 3SAT instance I', and that instance I has c clauses. Suppose furthermore that for *any* truth assignment to I, at most $c-2$ clauses of I are satisfied. Which of the following statements must be true? Choose one.

○ Some truth assignment for I' may satisfy all clauses of I'.

● For any truth assignment for I', at least one clause of I' is not satisfied.

4. (1 point) Consider the decision problems:

UNSAT: An instance I is a Boolean formula that (just like in the worksheet on the reduction from SAT to 3SAT) is the conjunction ("and") of clauses, where each clause is the disjunction ("or") of literals. The instance is a Yes-instance if and only if I has *no* satisfying truth assignment.

3UNSAT: An instance I is an UNSAT instance that has exactly three literals per clause. The instance is a Yes-instance if and only if I has *no* satisfying truth assignment.

Is 3UNSAT $\leq_p$ UNSAT? That is, is there a polynomial time reduction that maps any instance I of 3UNSAT to an instance I' of UNSAT, such that I has no satisfying truth assignment if and only if I' has no satisfying truth assignment?

● Yes ○ No

Summary of the Transform-Instance algorithm from SAT instance I to 3SAT instance I':

- Replace each clause of I with one literal (l) by $(l \vee l \vee l)$ in I'
- Replace each clause of I with two literals $(l_1 \vee l_2)$ by $(l_1 \vee l_2 \vee l_2)$ in I'
- Each clause of I with three literals is also in I'
- Replace each clause of I of the form $(l_1 \vee l_2 \vee \ldots \vee l_k)$ with $k > 3$ literals by clauses

$$(l_1 \vee l_2 \vee x_{i+1}) \wedge (\bar{x}_{i+1} \vee l_3 \vee x_{i+2}) \wedge (\bar{x}_{i+2} \vee l_4 \vee x_{i+3}) \wedge \ldots$$
$$\wedge\ (\bar{x}_{i+(j-2)} \vee l_j \vee x_{i+(j-1)})\ \wedge \ldots$$
$$\ldots \wedge\ (\bar{x}_{i+(k-4)} \vee l_{k-2} \vee x_{i+(k-3)}) \wedge (\bar{x}_{i+(k-3)} \vee l_{k-1} \vee l_k)$$

This page intentionally left (almost) blank.

If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.

This page intentionally left (almost) blank.

If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.