NOTE TO STUDENTS: This file contains sample solutions to the term test together with the marking scheme and comments for each question. Please read the solutions and the marking schemes and comments carefully. Make sure that you understand why the solutions given here are correct, that you understand the mistakes that you made (if any), and that you understand *why* your mistakes were mistakes.

Remember that although you may not agree completely with the marking scheme given here it was followed the same way for all students. We will remark your test only if you clearly demonstrate that the marking scheme was not followed correctly.

For all remarking requests, please submit your request **in writing** directly to your instructor. For all other questions, please don't hesitate to ask your instructor during office hours or by e-mail.

Question 1. [12 MARKS]

Consider the following *Art Gallery Guarding* problem. We are given a set $\{p_1, p_2, \ldots, p_n\}$ of positive real numbers that specify the positions of paintings in a long hallway in an art gallery. We want to position guards in the hallway to protect every painting, using as few guards as possible. Suppose that a single guard can protect all the paintings within distance 1 or less of his or her position (on both sides).

We propose to solve this problem using the following greedy algorithm.

```
Sort the positions so p_1 ⩽ p_2 ⩽ ··· ⩽ p_n.
G ← ∅    # current set of guards' locations
g ← −∞    # position of rightmost guard in G = maximum number in G
for i ← 1,...,n:    # start at the leftmost painting and move to the right
    if p_i > g + 1:    # p_i is unprotected by the guards currently in G
        # Place a guard 1 unit to the right of p_i.
        g ← p_i + 1
        G = G ∪ {g}
return G
```

Write a detailed proof that this algorithm always finds a minimum number of guards. We recommend that you follow the proof format outlined in class: start with a clear definition of "promising partial solution" for the algorithm, then use this as part of your argument.

HINT: You may want to start with an outline and fill in the details later—this is likely to be a bit long.

SAMPLE SOLUTION:

Let $G_0, G_1, \ldots, G_n$ be the partial solutions generated by the algorithm. Say G_i is "promising" iff there is some solution G_i^* such that:

- $G_i \subseteq G_i^*$ (G_i^* contains every guard position in G_i);
- $\forall g \in G_i^* - G_i, g > g_i = \max\{g' \in G_i\}$ (every additional guard in G_i^* is positioned further right than the last guard in G_i)—we say G_i^* "extenda" G_i if it satisfies the first two properties;
- $|G_i^*|$ is minimum (G_i^* is optimum).

CLAIM: $\forall i \geqslant 0$, G_i is promising.

PROOF: By induction on i.

Base Case: $G_0 = \emptyset$. Let G^* be any optimum solution. Then $G_0 \subseteq G^*$ and $\forall g \in G^*, g > g_0 = -\infty$.

Ind. Hyp.: Suppose $i \geqslant 0$ and G_i^* extends G_i.

Ind. Step: Either $G_{i+1} = G_i$ or $G_{i+1} = G_i \cup \{p_{i+1} + 1\}$.

Case 1: If $G_{i+1} = G_i$, then $G_{i+1} = G_i \subseteq G_i^*$ and $\forall g \in G_i^* - G_{i+1}, g \in G_i^* - G_i \Rightarrow g > g_i = g_{i+1}$. So G_i^* already extends G_{i+1}.

Case 2: If $G_{i+1} = G_i \cup \{p_{i+1} + 1\}$, then either $p_{i+1} + 1 \in G_i^*$ or $p_{i+1} + 1 \notin G_i^*$.

Subcase A: If $p_{i+1} + 1 \in G_i^*$ then G_i^* already extends G_{i+1}: $G_{i+1} = G_i \cup \{p_{i+1} + 1\} \subseteq G_i^*$ and $\forall g \in G_i^* - G_{i+1}, g > g_i \wedge g \neq p_{i+1} + 1 \Rightarrow g > g_{i+1} = p_{i+1} + 1$.

Subcase B: If $p_{i+1} + 1 \notin G_i^*$ then $p_{i+1} > g_i + 1$ (from the algorithm) so p_{i+1} is unprotected by any guard in G_i. This means there is some $g \in G_i^* - G_i$ with $g > g_i$ (by the I.H.) and $g \leqslant p_{i+1} + 1$ (else p_{i+1} would be unprotected by G_i^*). Then, every painting protected only by g is also protected by $p_{i+1} + 1$. So $G_{i+1}^* = G_i^* - \{g\} \cup \{pi + 1 + 1\}$ is an optimum solution that extends G_{i+1}.

Hence, G_i is promising for $i = 0, 1, \ldots, n$. In particular, G_n is promising: $G_n \subseteq G_n^*$ and $\forall g \in G_n^* - G_n, g > g_n$, for some optimum G_n^*. But this means $G_n = G_n^*$, as desired.

MARKING SCHEME:

- [4 marks] **Structure:** clear attempt to define "promising," to give an inductive proof, to use an exchange lemma, and to argue the final solution is optimum—even if these are done incorrectly
- [3 marks] **Idea:** answer contains the argument that any other solution uses guard positions that are not as far right as the greedy solution, and can be replaced by the greedy positions—even if this is poorly structured or written up
- [5 marks] **Details:** correct definition of "promising," appropriate cases and sub-cases in the inductive proof, correct exchange lemma, applied at the appropriate place, and correct argument for the conclusion

MARKING CODES AND COMMENTS:

- **DA:** "Dependent on Algorithm"—many students assumed a specific optimal solution was generated by the algorithm (it does not have to be), which made it impossible to generalize the exchange argument.
- **ID:** Incomplete definition
- **WD:** Wrong definition
- **MC:** Missing Cases
- **NE:** No explanation
- **ND:** No definition
- **F1:** Fixing the positions of the optimal solutions: the positions of the guards in the optimal solution are not restricted to be 1 more than the positions of the paintings.

- **I1:** Did not use the main idea of the exchange lemma
- **E:** No exchange lemma
- **U1:** undefined term
- **G:** Without justification, asserting that if the algorithm does not add a guard, that it can be extended to the same optimal solution.

Question 2. [10 MARKS]

A new "healthy fast-food" restaurant sells *Tofu Nuggets* in boxes that contain either 3, 10 or 25 nuggets. You visit the restaurant with a group of your friends. Can you purchase exactly the right amount of nuggets so that everyone gets to try one but nobody has to try more than one?

We formalize the "Tofu Nugget" problem as follows.

- **Input:** An integer $N \geqslant 0$ (the size of your group).
- **Question:** Determine whether or not you can purchase exactly N Tofu Nuggets (given that you can only purchase boxes of 3, 10 or 25).

Write an algorithm that solves this problem by using dymamic programming, and analyse its running time. Follow the steps from the "dynamic programming paradigm" covered in class. In particular, describe carefully the recursive structure of the problem in order to justify the correctness of your recurrence.

HINT: There is nothing to optimize—use an array that simply stores whether or not a solution exists.

SAMPLE SOLUTION:

Recursive Structure: If $N = 3x + 10y + 25z$ for some $x, y, z \in \mathbb{N}$, then either $x > 0$ and $N = 3 + 3(x-1) + 10y + 25z$, or $y > 0$ and $N = 10 + 3x + 10(y-1) + 25z$, or $z > 0$ and $N = 25 + 3x + 10y + 25(z-1)$.

Array: For $i = -24, -23, \ldots, -1, 0, 1, \ldots, N$, $A[i]$ = True if i nuggets can be purchased exactly (False otherwise).

Recurrence: $A[-24] = \cdots = A[-1] =$ FALSE
$A[0] =$ TRUE ($0 = 3 \cdot 0 + 10 \cdot 0 + 25 \cdot 0$)
$A[i] = A[i-3] \vee A[i-10] \vee A[i-25]$, for $i = 1, \ldots, N$ (from argument above)

Algorithm:

```
for i = -24,...,-1:
    A[i] ← False
A[0] ← True
for i = 1,...,N:
    A[i] = A[i-3] ∨ A[i-10] ∨ A[i-25]
return A[N]
```

Runtime is $\Theta(N + 25)$.

MARKING SCHEME:

- [3 marks] **Structure:** clear attempt to define an array indexed by subproblems, to give a recurrence for the array values (with justification), and to write an iterative algorithm based on the recurrence —even if these are done incorrectly
- [3 marks] **Idea:** answer contains the argument that one of the box sizes must be used in any solution, so we can simply try each one in turn—even if this is poorly structured or written up
- [4 marks] **Details:** correct array definition, correct recurrence with appropriate base cases and justification, correct algorithm (even if based on wrong recurrence), and correct runtime (even if algorithm is wrong)

MARKING CODES AND COMMENTS:

- **G:** this is a greedy approach and will not work
- **R:** runtime was not analysed
- **2D:** two or more parameters in your array—not necessary (see the hint)
- **N:** define what happens for array lookups at negative indices

Question 3. [8 MARKS]

Let $N = (V, E)$ be any network with integer capacities. Let F be the maximum flow value in N. Let P be a simple path in N from s to t and let N' be the network obtained from N by adding 1 to the capacity of every edge on P.

Part (a) [4 MARKS]

True or False? The maximum flow value in N' is *at least* $F + 1$.
Justify your answer (give a short proof or a counterexample).

SAMPLE SOLUTION:

> True. Augment f along path P: since each edge of P has residual capacity at least 1 in N', $F' \geqslant F + 1$.

MARKING SCHEME:

- [1 mark] correct answer (True/False)
- [3 marks] good justification

MARKING CODES AND COMMENTS:

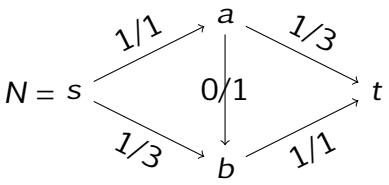
- **E1:** vague/ambiguous explanation: no reference to augmentation or to path P
- **E2:** incorrect explanation based on increased capacity of **one** edge only

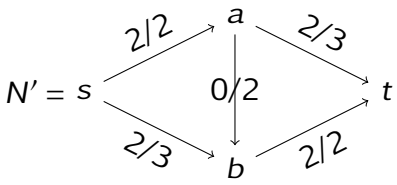
Part (b) [4 MARKS]

True or False? The maximum flow value in N' is *exactly* $F + 1$.
Justify your answer (give a short proof or a counterexample).

SAMPLE SOLUTION:

> False. Consider network N below with maximum flow value $F = 2$. Consider path $P = s \to a \to b \to t$ in N. Maximum flow value in N' is $F' = 4 > F + 1$.





MARKING SCHEME:

- [1 mark] correct answer (True/False)
- [3 marks] good justification (can get up to 2 marks for reasonable attempt to prove True)

MARKING CODES AND COMMENTS:

- **E1:** attempt to prove True by saying P crosses "the" min-cut once
- **E2:** attempt to prove True with an example!
- **E3:** attempt to prove True by arguing only one bottleneck/critical edge is affected
- **E4:** incorrect counterexample
- **E5:** argument involving only the first edge in P
- **E6:** assuming **every** edge capacity has increased by 1—*many* students misread the question and made this mistake