CS4311
Design and Analysis of Algorithms

Tutorial: Assignment 3 Hints
Question 1

• A stair has \( n \) steps
• Each time, Jack can walk up 1, 2, or 3 steps
• How many ways can Jack walk up the stair?

• Dynamic Programming
Question 2

- 2 piles of coins
- 2 players take turn to get coins
- Each turn, can either get
  (1) any #coins from one pile, or
  (2) same #coins from both piles
- Lose if does not get any coin in his turn
Question 2

- Assume both players are clever
  ➔ some combination are losing

E.g., If initially, the piles contain 1 coin and 2 coins, respectively, then we have only four ways of getting coin in this turn ...

(What are they?)
Question 2

Let \((x,y)\) denote the status with \(x\) coins and \(y\) coins in the piles.

- 4 ways to get coins from \((1,2)\):
  - Case 1: \((1,2) \rightarrow (0,2)\)
  - Case 2: \((1,2) \rightarrow (1,1)\)
  - Case 3: \((1,2) \rightarrow (1,0)\)
  - Case 4: \((1,2) \rightarrow (0,1)\)

\(\Rightarrow\) All are losing \(\Rightarrow\) \((1,2)\) is losing
Question 2

• In general, can we determine if \((x, y)\) is a losing combination?

• Dynamic Programming
Question 3

• John has a car, which can travel $n$ km when gas tank is full
• $k$ gas stations between SF and Seattle
• How to stop for fewest # of stations?

• Greedy Algorithm
Question 4

- If a Min-Heap contains $n$ elements
  - $\text{Extract-Min}$ takes $O(\log n)$ time
  - $\text{Insert}$ takes $O(\log n)$ time

- Design potential function so that:
  - $\text{Extract-Min} : O(1)$ amortized time
  - $\text{Insert} : O(\log n)$ amortized time
Question 4

• Extract-Min deletes some leaf node
• Each node should store some potential to prepare for its future deletion
• Potential function can be defined as the sum of the potentials in each node

• Question:
What potential should we store in a node?

[There are also other ways to assign the potential]
Question 5

• A sorted array supports fast searching, but slow insertion
• Can we trade searching time for insertion time?

• Our Scheme:
  Partition $n$ elements based on its binary representation of $n$
Question 5

• Let \( k = \lceil \log (n+1) \rceil \), and

\[
\langle b_{k-1}, b_{k-2}, \ldots, b_2, b_1, b_0 \rangle
\]

be binary representation of \( n \)

• Partition \( n \) elements to \( k \) sorted arrays such that:
  • if \( b_j = 1 \), array \( A_j \) holds \( 2^j \) elements
  • if \( b_j = 0 \), array \( A_j \) is empty
Question 5

For example,

- when $n = 5$, we have $5_{(dec)} = 101_{(bin)}$
  - two non-empty arrays, one with 4 elements, one with 1 element

- when $n = 11$, we have $11_{(dec)} = 1011_{(bin)}$
  - three non-empty arrays, with 8, 2, 1 elements, respectively
Question 5

- **Searching time:** $O(k \log n) = O(\log^2 n)$
- **Insertion time:**
  - Have to change the partitioning & to make sure each array is sorted
  - How to do so with amortized insertion time = $O(\log n)$?

- **Aggregate Method** should be the easiest among the three methods
Question 6 (Bonus)

• An extension to Question 2
• We show a method to generate losing combinations:

Set $L_0 = (0,0)$

for $k = 1,2, \ldots$ {
    Set $v = \text{smallest unseen positive #}$;
    Set $L_k = (v, v+k)$;
}

Question 6 (Bonus)

$L_0 = (0,0)$

for $k = 1, 2, \ldots$ {
    $v = \text{smallest unseen positive}$
    $L_k = (v, v+k)$;
}

Sample Run:

$L_0 = (0,0)$
$L_1 = (1,2)$
$L_2 = (3,5)$
$L_3 = (4,7)$
$L_4 = (6,10)$
$L_5 = (8,13)$

Interesting but unrelated fact:
This function generates each positive # exactly once
Question 6 (Bonus)

- Part (a) and (b):
  - properties about losing combinations

- Part (c):
  - Only losing combinations are generated

- Part (d):
  - a losing combination containing \( x \) must be generated (in \( O(x) \) steps)

- Part (e) and (f): running time = \( O(n) \)