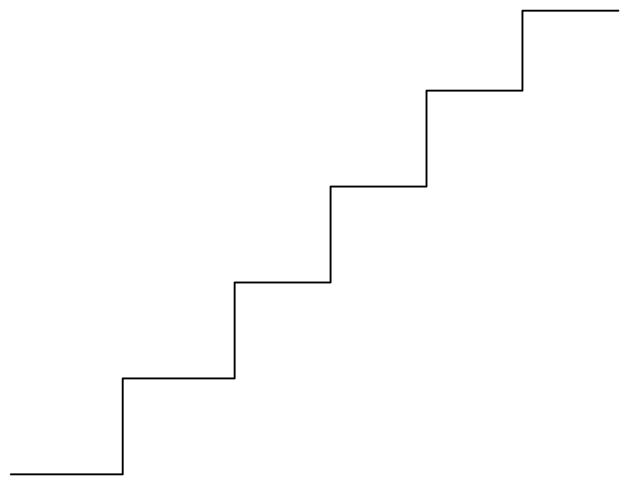
CS4311
Design and Analysis of Algorithms

Tutorial: Assignment 3 Solution

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 1

Let F_k = #ways to walk k steps

- Recurrence:

$$F_1 = 1\ ,\quad F_2 = 2,\quad F_3 = 4$$

$$F_{k+3} = F_k + F_{k+1} + F_{k+2}$$

- With DP, we can find each F_k in O(1) time
  ➔ total O(n) time to find F_n

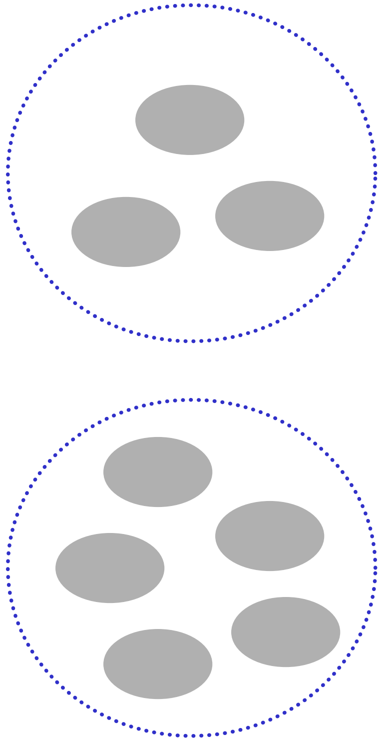
Remark: Using matrix multiplication, we can find F_n in O(log n) time

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

- Let

$$A(x,y) = L$$

if (x,y) is a losing combination,
and

$$A(x,y) = W$$

otherwise

Question 2

- Recurrence:
  - $A(0,0) = L$
  - $A(i,j) = W$

    if $A(s, j) = L$ for some $0 \le s < i$

    or $A(i, t) = L$ for some $0 \le t < j$

    or $A(i-k, j-k) = L$ for some $1 \le k \le \min(i,j)$
  - $A(i,j) = L$ otherwise

Reason: A combination is losing if and only if every move yields a winning combination (for your opponent)

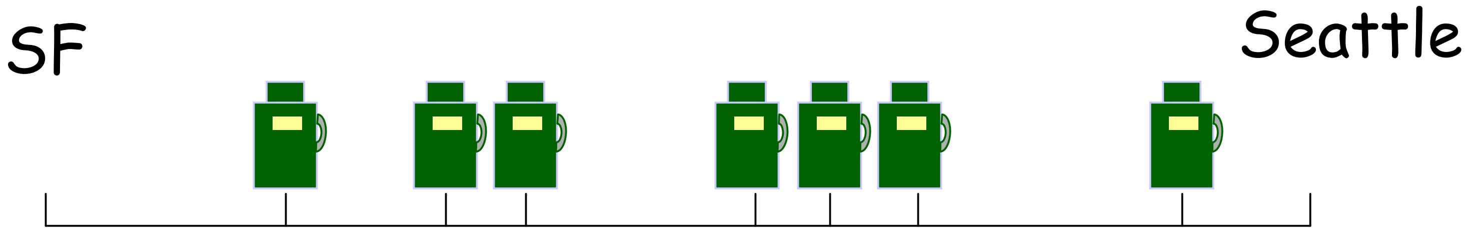
Question 2

- With DP, we can find each $A(i,j)$ in $O(n)$ time (based on $O(n)$ previously filled entries)

- To find $A(x,y)$, there are $x \times y = O(n^2)$ entries
  ➔ total $O(n^3)$ time

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 3

- Let s = rightmost (farthest) gas station within the first n km from SF

Greedy Choice Lemma:

> There is an optimal solution (with fewest #stations) whose leftmost station is s

Proof: (By cut-and-paste)

- If s ≠ leftmost, we can replace leftmost one by s → a feasible solution
- #stations cannot be increased → optimal

Question 3

Let $S(x,y)$ = optimal #stations to travel from x to y (starting with a full-tank at x)

Optimal Substructure Lemma:

If an optimal solution to travel from x to y uses station s, then

$$1 + S(x,s) + S(s,y)$$

Proof: (By contradiction)

Question 3

- The lemmas imply this greedy algorithm :

```
1.  Choose s_1 =  rightmost gas station from
                  SF within first n km ;
2.  k = 1 ;
3. while ( distance(s_k, Seattle) > n )  {
      Choose s_{k+1} =  rightmost gas station
                        from s_k within first n km ;
      k = k + 1 ;
   }
```

Question 4

- If a Min-Heap contains n elements
  - Extract-Min takes $O(\log n)$ time
  - Insert takes $O(\log n)$ time
- Design potential function so that:
  - Extract-Min : $O(1)$ amortized time
  - Insert : $O(\log n)$ amortized time

Question 4

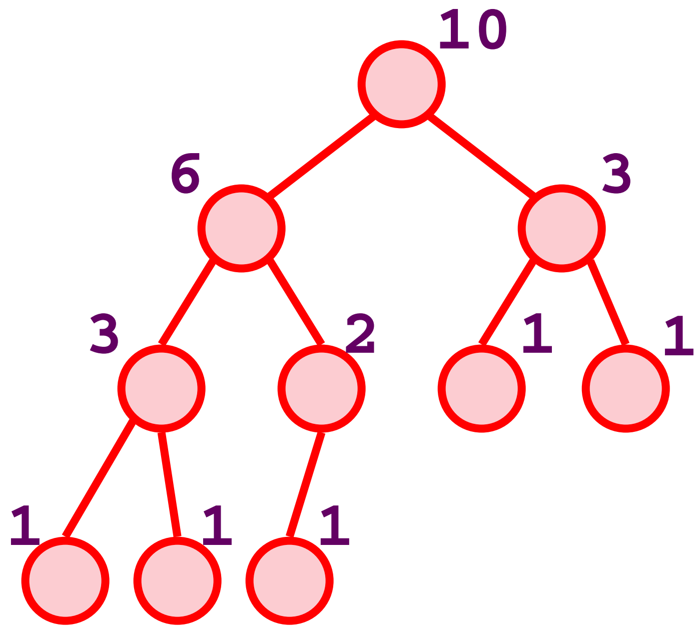
Solution 1:

For each node u,

$\Phi(u)$ = size of subtree rooted at u

Potential of a heap H :

$\Phi(H)$ = sum of potentials of all nodes



Question 4

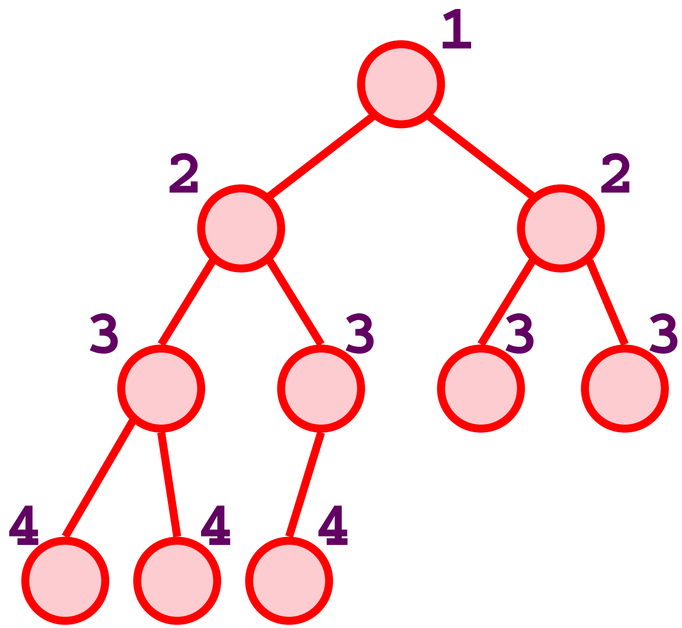
Solution 2:

For each node u,

$\Phi(u)$ = node-depth of u

Potential of a heap H :

$\Phi(H)$ = sum of potentials of all nodes



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

- When binary representation is in the form:

$$\langle\, ?, ?, 0, \underbrace{1, 1, \ldots, 1}_{r \text{ one's}} \,\rangle$$

  - A further insertion will increase #elements by 1, so that the new binary representation becomes:

$$\langle\, ?, ?, 1, \underbrace{0, 0, \ldots, 0}_{r \text{ zero's}} \,\rangle$$

Question 5

- In this case, we merge r sorted lists + new element (total 2^r elements) into one sorted list
- Total time : $O(1+2+4+\ldots+2^r) = O(2^r)$ time
- For m inserts,
  #times we merge r sorted lists = $O(m/2^r)$

Since r ranges from 0 to log n,

Total insertion: $\sum_r 2^r O(m/2^r) = O(m \log n)$ time

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, … {

  Set v = smallest unseen positive # ;

  Set L_k = (v, v+k) ;

}

Question 6 (Bonus)

$L_0 = (0,0)$
for $k = 1,2, \ldots$ {
  v = smallest unseen postive # ;
  $L_k = (v, v+k)$;
}

Interesting but unrelated fact:
This function generates each positive # exactly once

Sample Run:

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

Question 6 (Bonus)

6(a). At most one x with $(x,x+k)$ losing :

Proof:

If on contrary, there are distinct x and y with $(x,x+k)$ and $(y,y+k)$ both losing, we can transform one to another in one move (taking same # in both piles) → contradiction

6(b). At most one r with (x,r) losing :

Proof: Similar to 6(a)

Question 6 (Bonus)

6(c). Each $L_k = (v,v+k)$ generated is losing :

Proof: We show that each move yields a winning combination (for the opponent)

Case 1: Taking in first pile:

→ By 6(b), $(v',v+k)$ must be winning (why?)

Case 2: Taking from both piles:

→ By 6(b), $(v',v'+k)$ must be winning (why?)

Case 3: next slide

Question 6 (Bonus)

6(c). Each $L_k = (v, v+k)$ generated is losing :

Proof: …

Case 3: Taking from second pile:

(I) Taking at most k:

By 6(a), $(v, v+k')$ must be winning

(II) Taking more than k:

By 6(b), (v, v') must be winning

Question 6 (Bonus)

6(d). After x iterations, L_i must contain x

Proof:

The v value increases by at least one after each iteration → v value of L_x is at least x

6(e). Total time for x iterations = $O(n)$

- The most time consuming step is to find the smallest unseen integer
- Use array of size $O(n)$ → $O(n)$ time

Question 6 (Bonus)

6(f). Decide (x,y) is losing takes $O(n)$ time

Once all L_0 to L_x are computed, we can find the entry containing x

→ By 6(b), we can determine if (x,y) is losing in extra $O(1)$ time