CS 460, HOMEWORK 2
(DUE: OCT 8, 11:59 PM)

INSTRUCTOR: HOA VU

- Each question is worth 20 points.
- Submission should be in PDF. There are some scanner apps that improves the readability if you handwrite the solutions.
- You can work in groups of 2. Make sure to list the names of people in your group. Each student must still submit a copy on Canvas.
- When you are asked to design an algorithm, do the following: a) describe the algorithm, b) prove/argue why it is correct, and c) provide the running time.
- [Erickson] denotes the book by Jeff Erickson (available for free online at `http://jeffe.cs.illinois.edu/teaching/algorithms/`).
- [DPV] denotes the book by Dasgupta, Papadimitriou, and Vazirani (the required textbook).

(1) **Question 1:** A sorted array $A[1 \dots n]$ is circular-shifted k position where k is unknown. Find k in $O(\log n)$ time. For example. if the input is

$$A = [23, 25, 1, 10, 17]$$

Then, the algorithm should output $k = 2$.

(2) **Question 2:** Problem 6.1 of [DPV].

(3) **Question 3:** Problem 6.17 of [DPV].

(4) **Question 4:** Problem 6.11 of [DPV].

(5) **Question 5:** Implement the coin changing problem on Leetcode. Follow the following link `https://leetcode.com/problems/coin-change/`. You need to do the following: 1) provide your code in the homework submission, 2) after your code successfully passed all the test cases when you click on the "submit" button, click on the "details" button on the top left, you will see a report of the performance of your code (which would look similar to the picture below). Include that in your submission.

Longest Common Subsequence

Submission Detail

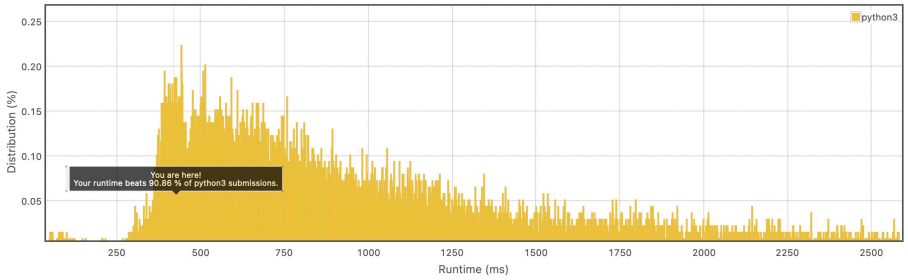
45 / 45 test cases passed.

Runtime: **422 ms**
Memory Usage: **22.7 MB**

Status: **Accepted**

Submitted: **6 minutes ago**

Accepted Solutions Runtime Distribution



(1) **Question 1:** A sorted array $A[1 \ldots n]$ is circular-shifted k position where k is unknown. Find k in $O(\log n)$ time. For example. if the input is

$$A = [23, 25, 1, 10, 17]$$

Then, the algorithm should output $k = 2$.

- Firstly, we assume the numbers of input array are all distinct. (If the input contains identical numbers, we can't find any algorithm solving this problem in the $O(\log n)$)

1. Divide the input array into 2 parts : A_{left} and A_{Right}.
2. Compare $A_{left}[1]$ vs $A_{right}[1]$
   - If $A_{left}[1] > A_{right}[1]$ ⟶ the biggest number is in the A_{left}.
   - If $A_{left}[1] < A_{right}[1]$ ⟶ the biggest number is in the A_{right}.
3. Then we divide $A_{left/right}$ into 2 parts, find the biggest number recursively until only one number in the array (base case).

Divide the array into two smaller arrays and compare once per each iteration.

↳ Running time = $O(\log n)$.

6.1. A *contiguous subsequence* of a list S is a subsequence made up of consecutive elements of S. For instance, if S is

$$5, 15, -30, 10, -5, 40, 10,$$

then $15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is not. Give a linear-time algorithm for the following task:

Input: A list of numbers, $a_1, a_2, \ldots, a_n$.
Output: The contiguous subsequence of maximum sum (a subsequence of length zero has sum zero).

For the preceding example, the answer would be $10, -5, 40, 10$, with a sum of 55.

(*Hint:* For each $j \in \{1, 2, \ldots, n\}$, consider contiguous subsequences ending exactly at position j.)

Let $S[i]$ is the maximum sum of a contiguous subsequence that ends at x_i.

There are two cases:

① If the maximum contiguous sum consists only x_i then $S[i] = x_i$.

② Else, it has to be the maximum contiguous sum ends at x_{i-1} and x_i. ⟶ $S[i] = S[i-1] + x_i$.

Hence, $S[i] = \max\{S[i-1] + x_i, x_i\}$

▷Pseudocode:

```
S[1] = x_1
for i = 1, 2, 3, ..., n :
    S[i] = max{S[i-1] + x_i , x_i}
return max S[i].
```

] O(n)

6.17. Given an unlimited supply of coins of denominations $x_1, x_2, \ldots, x_n$, we wish to make change for a value v; that is, we wish to find a set of coins whose total value is v. This might not be possible: for instance, if the denominations are 5 and 10 then we can make change for 15 but not for 12. Give an $O(nv)$ dynamic-programming algorithm for the following problem.

Input: $x_1, \ldots, x_n; v$.
Question: Is it possible to make change for v using coins of denominations $x_1, \ldots, x_n$?

Let C[v] be a predicate which evaluates to true if it make change for v using available denominations $x_1, x_2, \ldots, x_n$.
We can write the expression such that

$$C[v] = \begin{cases} \text{True} & \text{- if it is possible to make change for } v \\ \text{False} & \text{, otherwise.} \end{cases}$$

Since it is possible to make change for v using the given denomination $x_1, x_2, \ldots, x_n$ then it is also possible to make change for $v - x_i$ with $x_i \le v$ by using the same denominations with one coin of x_i being chosen.

Recursive definition of C[i] is:

$$C[v] = \begin{cases} C[v - x_i] & \text{if } x_i \le v \\ \text{False}, & \text{otherwise} \end{cases} \qquad \forall\ 1 \le i \le n$$

Pseudocode:

```
Declare an array C of size v+1
C[0] = true
for i = 1, 2, ... , v
    C[i] = false
for i = 1, 2, ... , v:
    for j = 1, 2, 3, ... , n:                    ] O(n)  ] O(n·v)
        if xj ≤ v:
            C[v] = C[v] v C[v-xj]
        else :
            C[v] = false
return C[v].
```

Running time = $O(n \cdot v)$

6.11. Given two strings $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_m$, we wish to find the length of their *longest common subsequence*, that is, the largest k for which there are indices $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ with $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{j_1} y_{j_2} \cdots y_{j_k}$. Show how to do this in time $O(mn)$.

Let $Z[i,j]$ be the length of the longest common subsequence of two given strings $X = x_1 x_2 \dots x_n$ and $Y = y_1 y_2 \dots y_m$.

If $x_i = y_j$, then they will be in the common subsequence.

Else: we need to get the longer between the LCS of $(x_1 \dots x_{i-1})$ and $(y_1 \dots y_j)$ vs the LCS of $(x_1 \dots x_i)$ and $(y_1 \dots y_{j-1})$.

$$Z[i,j] = \begin{cases} Z[i-1,j-1] + 1 & \text{if } x_i = y_j \\ \max\{Z[i-1,j], Z[i,j-1]\} & \text{if } x_i \neq y_j \end{cases}$$

Initialize first row and column of $Z[i,j]$: $Z[0,j] = 0$ and $Z[i,0] = 0$

▷Pseudocode:

```
Z[0,j] = 0
Z[i,0] = 0
for i = 1,2,...,n:
    for j = 1,2,...,m:
        Z[i,j] = { Z[i-1,j-1] + 1          if x_i = y_j
                 { max{Z[i-1,j], Z[i,j-1]}  if x_i ≠ y_j.
```

O(n·m)

return $Z[n,m]$.

(5) **Question 5:** Implement the coin changing problem on Leetcode. Follow the following link `https://leetcode.com/problems/coin-change/`. You need to do the following: 1) provide your code in the homework submission, 2) after your code successfully passed all the test cases when you click on the "submit" button, click on the "details" button on the top left, you will see a report of the performance of your code (which would look similar to the picture below). Include that in your submission.

a. Code part:

```java
class Solution {
    public int coinChange(int[] coins, int amount){
        int [] answer = new int[amount + 1];
        for (int i = 1; i <= amount; i++){
            answer[i] = Integer.MAX_VALUE;
            for(int j = 0; j < coins.length; j++){
                if(coins[j] <= i && answer[i-coins[j]] != Integer.MAX_VALUE){
                    answer[i] = Integer.min(answer[i], 1+ answer[i-coins[j]]);
                }
            }
        }
        if(answer[amount]==Integer.MAX_VALUE){
            return -1;
        }
        return answer[amount];
    }

}
```

Back to problem

b. Report of the performance:

Submission Detail

189 / 189 test cases passed.

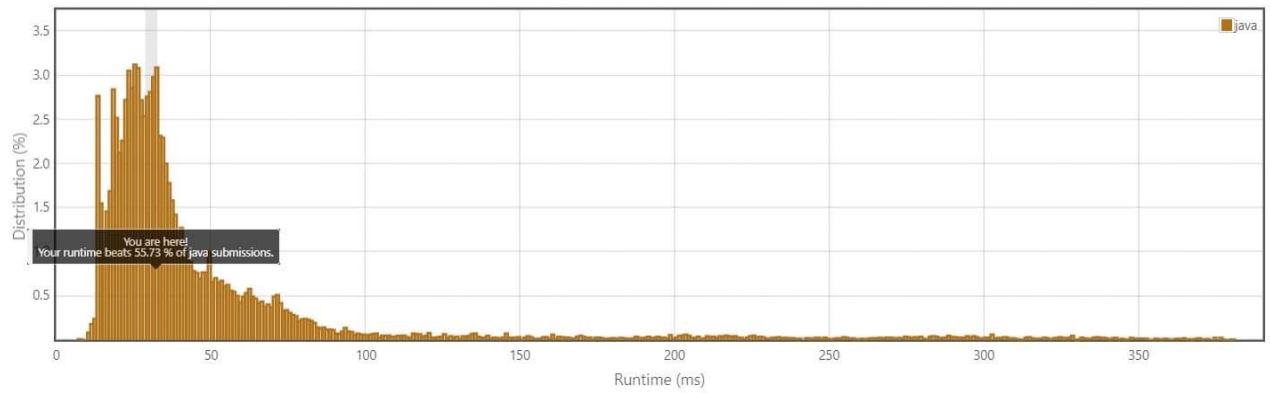
Runtime: **31 ms**

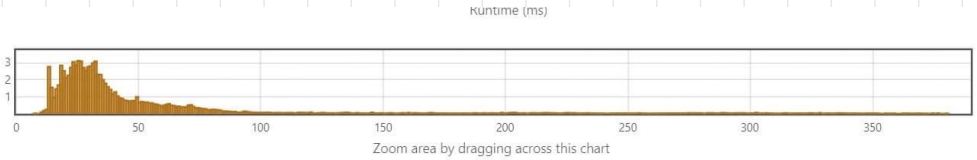
Memory Usage: **43.5 MB**

Status: **Accepted**

Submitted: **20 hours, 55 minutes ago**

Accepted Solutions Runtime Distribution





Accepted Solutions Memory Distribution