207. Course Schedule

DFS

```
from collections import defaultdict
class Solution:
    def canFinish(self, numCourses: int, prerequisites: List[List[int]]) -> bool:
        def dfs(cur):
            if visited[cur] == 1:
                return True
            elif visited[cur] == -1:
                return False
            visited[cur] = -1
            for i in memo[cur]:
                if not dfs(i):
                    return False
            visited[cur] = 1
            return True

        memo = defaultdict(set)
        visited = [0 for i in range(numCourses)]
        for s, e in prerequisites:
            memo[e].add(s)
        for i in range(numCourses):
            if not dfs(i):
                return False
        return True
```

Deck Cards OOD

```
from enum import Enum
from random import *


class SUITE(Enum):
   HEART = 1,
   SPADE = 2,
   CLUB = 3,
   DIAMOND = 4


class Card:
   def __init__(self, value, suite=SUITE.CLUB):
       self.value = value
       self.suite = suite

   def get_value(self):
       return self.value

   def set_value(self, value):
       self.value = value

   def get_suite(self):
       return self.suite

   def set_suite(self, suite):
       self.suite = suite



class Deck:
   def __init__(self):
       self.cardDeck = []
       for i in range(1, 14):
           self.cardDeck.append(Card(i, SUITE.CLUB))
           self.cardDeck.append(Card(i, SUITE.DIAMOND))
           self.cardDeck.append(Card(i, SUITE.HEART))
           self.cardDeck.append(Card(i, SUITE.SPADE))

   def shuffle(self):
       for _ in range(20):
           firstCard = randint(0, 51)
           secondCard = randint(0, 51)
           self.cardDeck[firstCard], self.cardDeck[secondCard] =
self.cardDeck[secondCard], self.cardDeck[firstCard]

   def dealCard(self):
       if not self.cardDeck:
           return None
       card = self.cardDeck.pop()
       return card

   def getSizeOfDeck(self):
       return len(self.cardDeck)



class Hand:
   def __init__(self):
       self.hand = []

   def getHand(self):
       return self.hand

   def getCard(self, card):
       self.hand.append(card)



hand1 = Hand()
hand2 = Hand()
deckCard = Deck()
deckCard.shuffle()
hand1.getCard(deckCard.dealCard())
hand2.getCard(deckCard.dealCard())
print(hand1.getHand())
print(hand2.getHand())
```

Object Diff:

```
function getDiff(obj1, obj2) {
 const dif = {};
 for (let i in obj1) {
   if (obj1[i] !== obj2[i]) {
     dif[i] =
       JSON.stringify({ i: obj1[i] }) + "|" + JSON.stringify({ i: obj2[i] });
   }
 }
 return dif;
}

function getOrigin(obj1, diff) {
 const obj2 = {};
 for (let i in obj1) {
   if (diff[i]) {
     const [str1, str2] = diff[i].split("|");
     if (JSON.stringify({ i: obj1[i] }) === str1) {
       obj2[i] = JSON.parse(str2)["i"];
     } else {
       obj2[i] = JSON.parse(str1)["i"];
     }
   } else {
     if (Array.isArray(obj1[i])) {
       obj2[i] = [...obj1[i]];
     } else if (typeof obj1[i] === "object") {
       obj2[i] = { ...obj1[i] };
     } else {
       obj2[i] = obj1[i];
     }
   }
 }
 return obj2;
}

const obj1 = { arr: [1, 2, 3], brr: "fuck", c: 123 };
const obj2 = { arr: [1, 2], brr: "fuck1", c: 1232 };
const diff = getDiff(obj1, obj2);
console.log(getOrigin(obj1, diff));
console.log(getOrigin(obj2, diff));
```

456. 132 Pattern
Using stack

```
class Solution:
    def find132pattern(self, nums: List[int]) -> bool:
        stack = []
        s3 = -float('inf')
        for num in reversed(nums):
            if s3 > num:
                return True
            while stack and stack[-1] < num:
                s3 = stack.pop()
            stack.append(num)
        return False
```

973. K Closest Points to Origin

```
from heapq import *
class Solution:
    def kClosest(self, points: List[List[int]], K: int) -> List[List[int]]:
        hp = []
        for point in points:
            if len(hp) < K:
                heappush(hp, (-point[0] ** 2 - point[1] ** 2, point))
            else:
                heappushpop(hp, (-point[0] ** 2 - point[1] ** 2, point))
        return list(map(lambda a:a[1], hp))
```