Hackerrank:

Nhóm 2:

```
def SubSetSum(A, sum, d):
    global count
    if sum == d:
        count += 1
    elif sum < d and len(A) > 0:
        SubSetSum(A[1:], sum, d)
        SubSetSum(A[1:], sum+A[0], d)

if __name__ == '__main__':
    [n, d] = map(int, input().split())
    A = [int(x) for x in input().split()]
    count = 0
    SubSetSum(A, 0, d)
    print(count)
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài.

Nhóm 3:

```
n, target = list(map(int, input().split()))
arr = [int(x) for x in input().split()]
count = 0

def FindSubsetSum(arr, sum, x):
    if sum == x:
        global count
        count += 1
        return
    elif sum > x or len(arr) == 0:
        return
    else:
        FindSubsetSum(arr[1:], sum, x)
        FindSubsetSum(arr[1:], sum + arr[0], x)

FindSubsetSum(arr, 0, target)
print(count)
count = 0
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài.

Nhóm 4:

```
def subset(lst, s, d, i):
    if s == d:
        return 1
    elif s > d or i == len(lst) or (s + sum(lst[i:]) < d):
        return 0
    else:
        return subset(lst, s, d, i + 1) + subset(lst, s + lst[i], d, i + 1)
n, d = map(int, input().split())
lst = list(map(int, input().split()))

result = subset(lst, 0, d, 0)
print(result)
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài.

Nhóm 5:

```
def sumSetSolve(lst,i,s,d):
    if s == d:
        return 1
    if s > d or i == len(lst):
        return 0
    return sumSetSolve(lst,i + 1,s,d) + sumSetSolve(lst,i + 1,s + lst[i],d)

n,d = map(int,input().split())
lst = list(map(int,input().split()))

result = sumSetSolve(lst,0,0,d)

print(result)
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài.

Nhóm 7:

```
class SubsetSumProblem:
    def __init__(self, size, sum, array):
        self.sizeOfArray = size
        self.givenSum = sum
        self.array = array
        self.ans = 0

    def combinationSum(self, currSum=0, currIndex=0):
        if (currSum > self.givenSum):
```

```
            return

        if (currSum == self.givenSum):
            self.ans += 1
            return

        for index in range(currIndex, self.sizeOfArray):
            currSum += self.array[index]
            self.combinationSum(currSum, index + 1)
            currSum -= self.array[index]

        return self.ans
size, sum = [int(element) for element in input().split()]
array = list([int(element) for element in input().split()])
solution = SubsetSumProblem(size, sum, array)
print(solution.combinationSum())
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài. Cần giải thích vòng for.

Nhóm 8:

```
def SubsetSum(set, n, sum):
    if (sum == 0):
        return 1
    if (n == 0):
        return 0
    if (sum < 0):
        return 0

    if (set[n - 1] > sum):
        return SubsetSum(set, n - 1, sum)

    return SubsetSum(set, n-1, sum) + SubsetSum(set, n-1, sum-set[n-1])


def take_ipt():
    inp = input()
    inp = inp.split()
    n = int(inp[0])
    d = int(inp[1])

    arr = input()
    arr = arr.split()
    tmp = []

    for x in arr:
        tmp.append(int(x))
    arr = tmp
    return n, d, arr
```

```
n, d, arr = take_ipt()

if d == 0:
    print(0)
else:
    print(SubsetSum(arr, n, d))
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài. Bài của nhóm dùng phương án trừ phần tử thay vì cộng.

Nhóm 9:

```
class SubsetSumProblem:
    def __init__(self, size, sum, array):
        self.sizeOfArray = size
        self.givenSum = sum
        self.array = array
        self.ans = 0

    def combinationSum(self, currSum=0, currIndex=0):
        if (currSum > self.givenSum):
            return

        if (currSum == self.givenSum):
            self.ans += 1
            return

        for index in range(currIndex, self.sizeOfArray):
            currSum += self.array[index]
            self.combinationSum(currSum, index + 1)
            currSum -= self.array[index]

        return self.ans


if __name__ == "__main__":
    size, sum = [int(element) for element in input().split()]
    array = list([int(element) for element in input().split()])
    solution = SubsetSumProblem(size, sum, array)
    print(solution.combinationSum())
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài. Cần giải thích vòng for.

Nhóm 10:

```
arr = []
a = int(input())
b = int(input())
count = 0
for i in range(a):
    x = int(input())
    arr.append(x)
for i in range(a):
    for j in range(1, a):
        if arr[i] + arr[j] == b:
            count = count+1
print(count)
```

Nhận xét: Sai ý tưởng, sai yêu cầu đề bài, sai nhập xuất.

Nhóm 12:

```
count = 0
n, target_sum = map(int, input().split())
a = list(map(int, input().split()))


def sum_sub_set(arr, sum1, target_sum):
    global count
    if sum1 == target_sum:
        count += 1
        return
    elif sum1 > target_sum or len(arr) == 0:
        return
    else:
        sum_sub_set(arr[1:], sum1, target_sum)
        sum_sub_set(arr[1:], sum1 + arr[0], target_sum)
sum_sub_set(a, 0, target_sum)
print(count)
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài.

Nhóm 13:

```
def SubsetSum(arr, cur_sum, i, sum, count, n):
    if cur_sum == sum:
        count[0] += 1
        return
    elif cur_sum > sum or i == n:
        return
    else:
        SubsetSum(arr, cur_sum + arr[i], i + 1, sum, count, n)

        SubsetSum(arr, cur_sum, i + 1, sum, count, n)


n, sum = [int(x) for x in input().split()]
arr = [int(x) for x in input().split()]

cur_sum = 0
count = [0]

SubsetSum(arr, cur_sum, 0, sum, count, n)
print(count[0])
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài. Giải thích tại sao dùng count = 0 mà dùng count = [0]

Nhóm 14:

```
[n, d] = map(int, input().split())
lst = [int(x) for x in input().split()]


def SubSetSum(lst, sum, d):
    global count
    if sum == d:
        count += 1
    elif sum < d and len(lst) > 0:
        SubSetSum(lst[1:], sum, d)
        SubSetSum(lst[1:], sum + lst[0], d)


count = 0
SubSetSum(lst, 0, d)
print(count)
```

Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài.

Nhóm 15:

```
def SubSetSum(A, sum, d):
    global count
    if sum == d:
        count += 1
    else:
        if sum > d or len(A) == 0:
            return
        else:
            SubSetSum(A[1:], sum, d)
            SubSetSum(A[1:], sum + A[0], d)


n, d = map(int, input().split())
A = [int(x) for x in input().split()]

sum = 0
count = 0
SubSetSum(A, sum, d)
print(count)
```

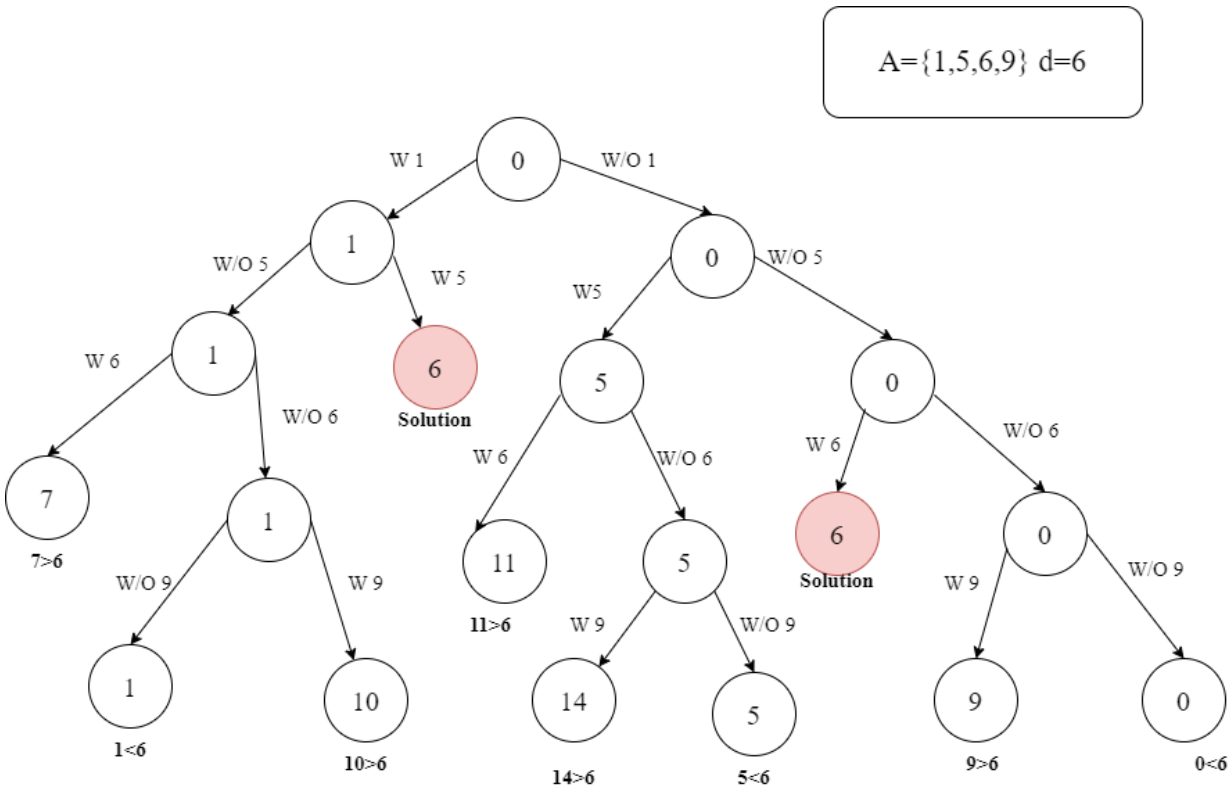
Nhận xét: Đúng ý tưởng, đạt điểm tối đa, đúng yêu cầu đề bài.

Nhóm 6: Không nộp bài

Nhóm 11: Không nộp bài

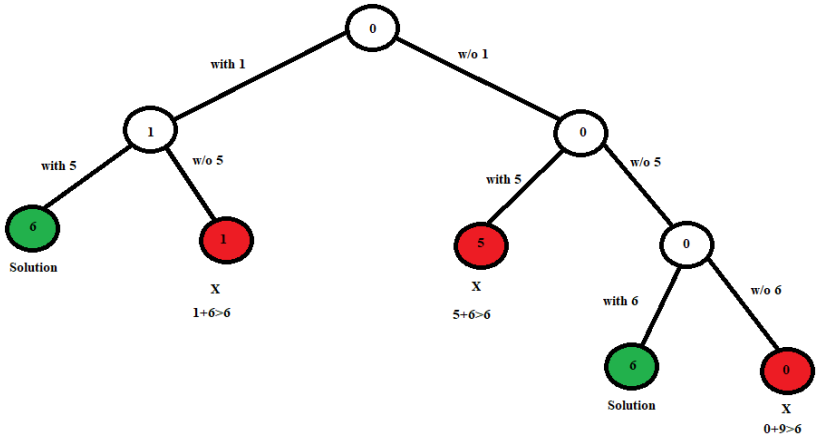
State – Space – Tree:

Nhóm 2:



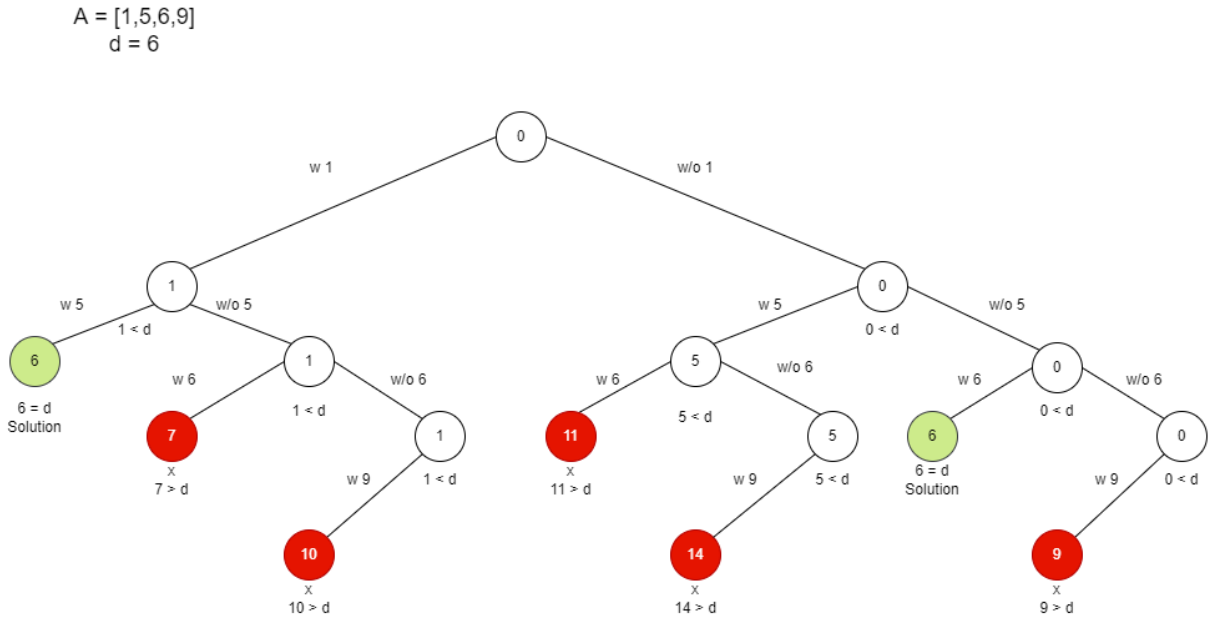
Nhận xét: Vẽ cây đúng và đầy đủ các trường hợp

Nhóm 3:



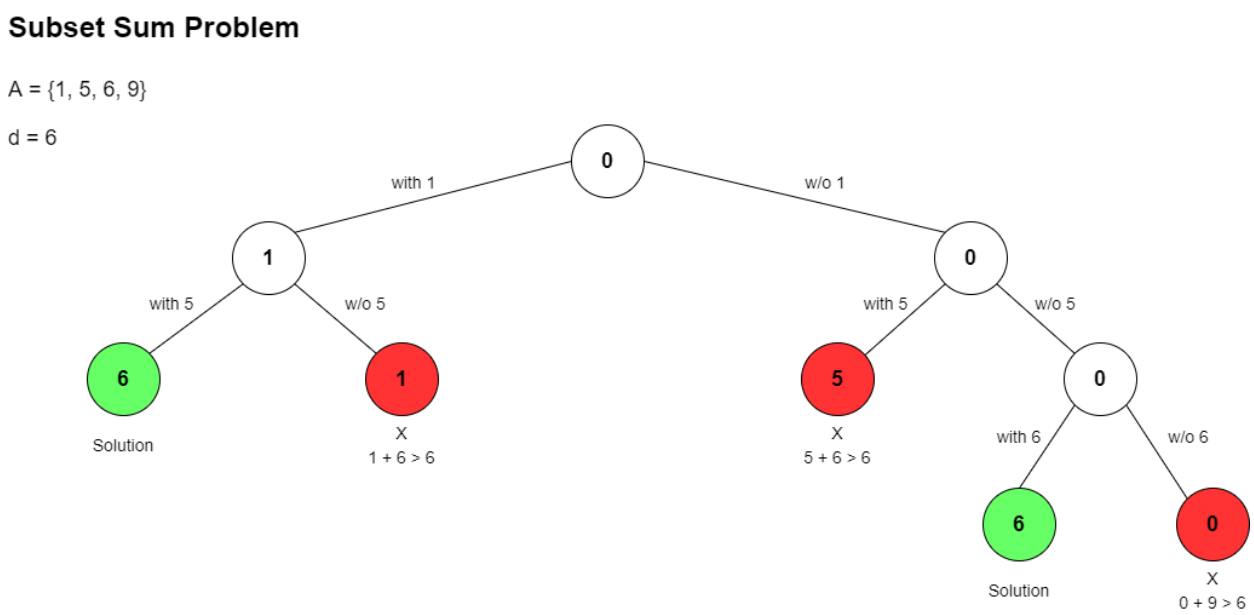
Nhận xét: nhóm vẽ thiếu nhiều trường hợp, có vẻ các bạn chưa hiểu được rõ ý tưởng của state-space tree.

Nhóm 4:



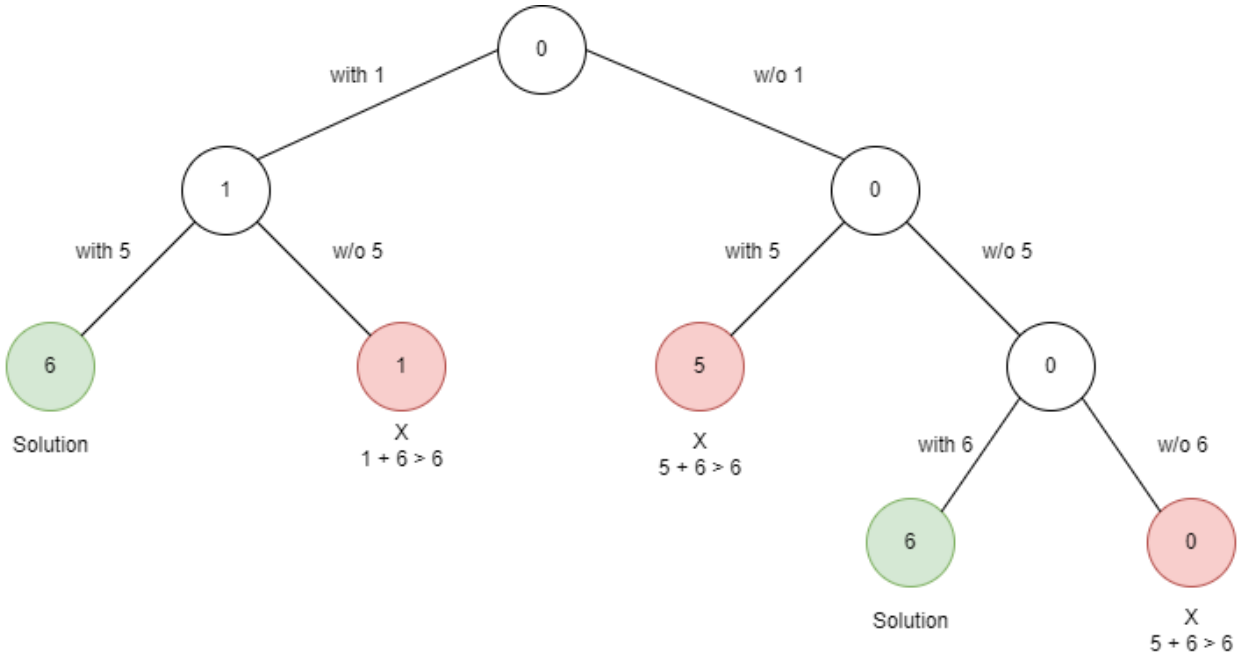
Nhận xét: Nhóm vẽ thiếu 1 vài trường hợp (5<d, 0<d,…)

Nhóm 5:



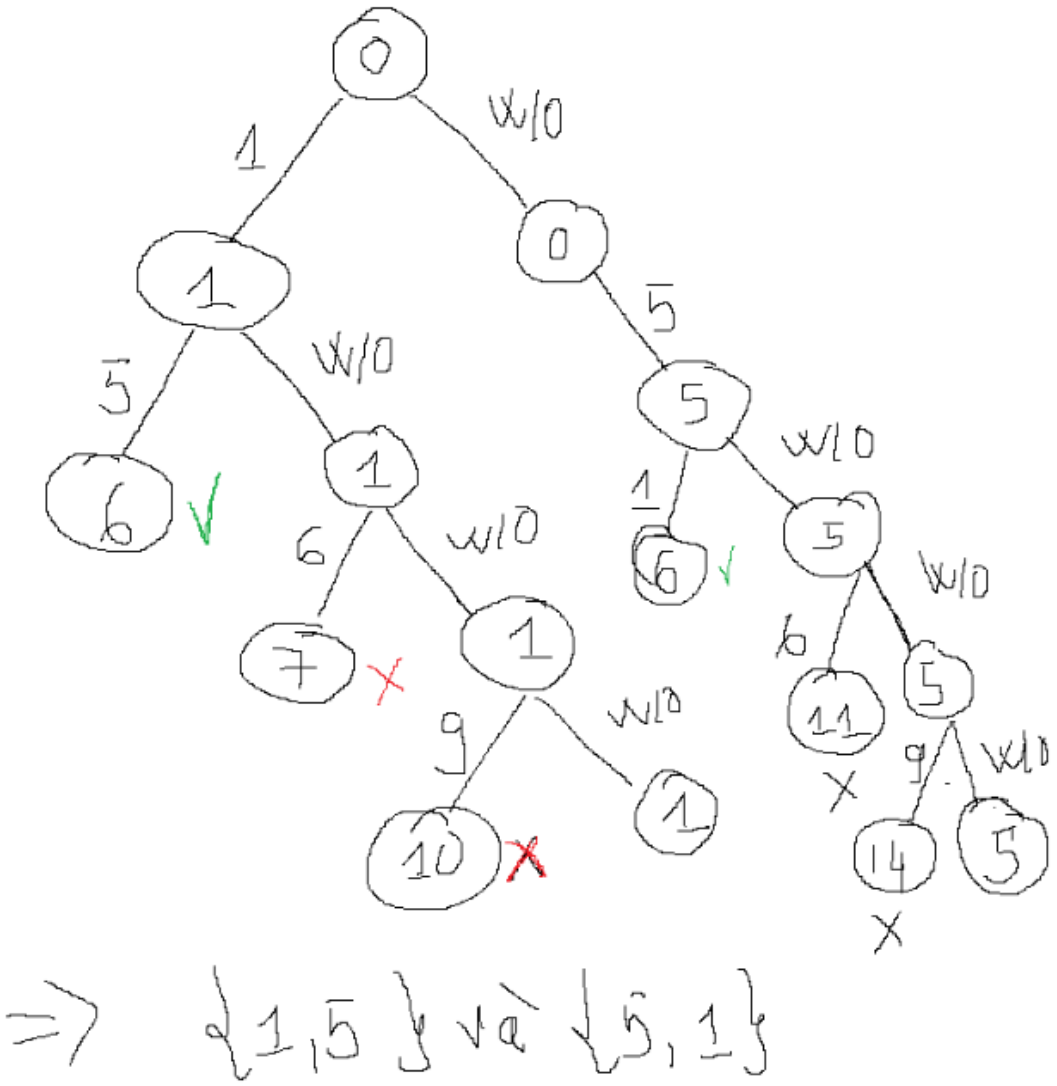
Nhận xét: Nhóm thiếu nhiều trường hợp, sai lầm giống nhóm 3

Nhóm 7:



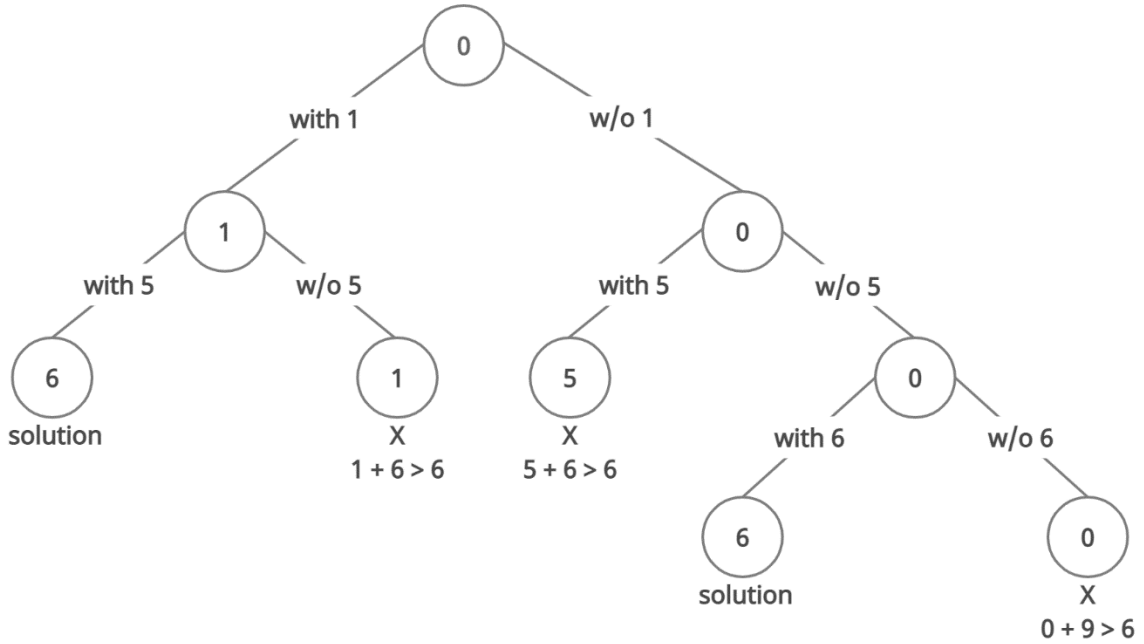
Nhận xét: Nhóm thiếu nhiều trường hợp, sai lầm giống nhóm 3

Nhóm 8:



Nhận xét: Nhóm vẽ sai state-space tree và thiếu trường hợp, cần phóng to hình ảnh khi nộp.

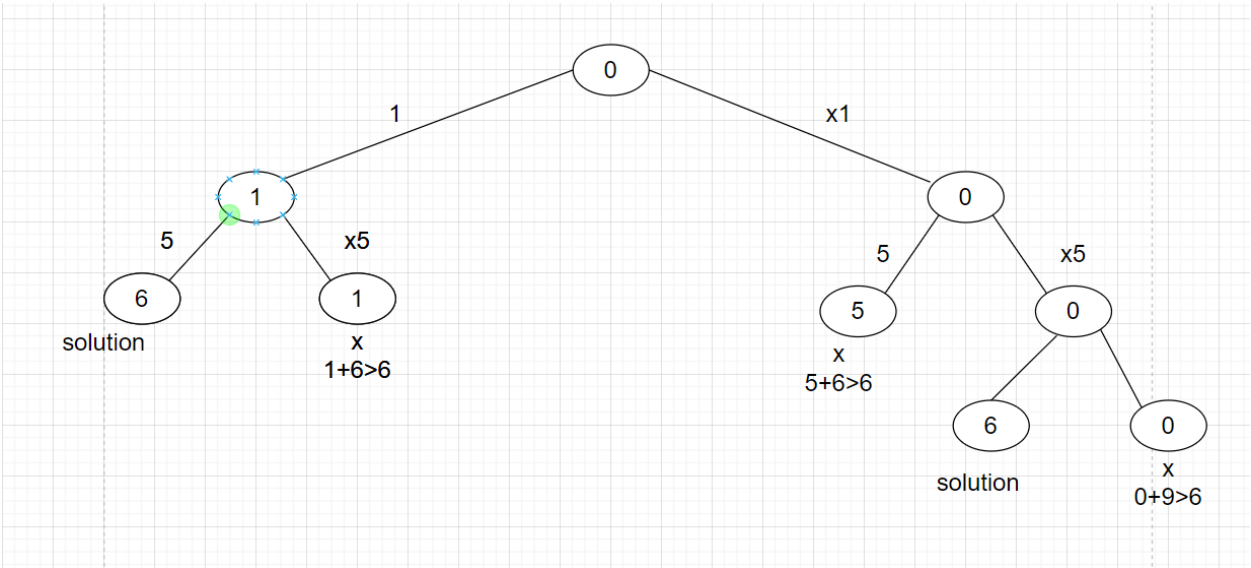
Nhóm 9:



Subset Sum Problem: Vẽ State-Space Tree cho trường hợp A = {1, 5, 6, 9}, d=6

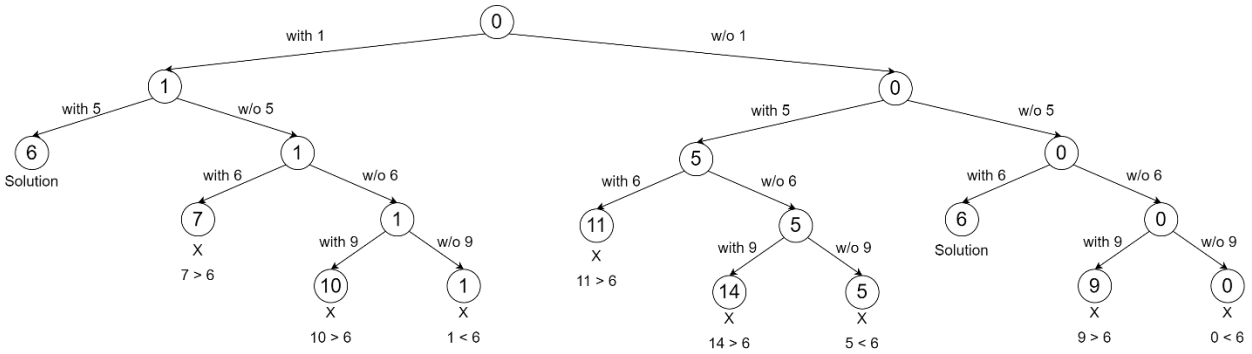
Nhận xét: Nhóm thiếu nhiều trường hợp, sai lầm giống nhóm 3

Nhóm 12:



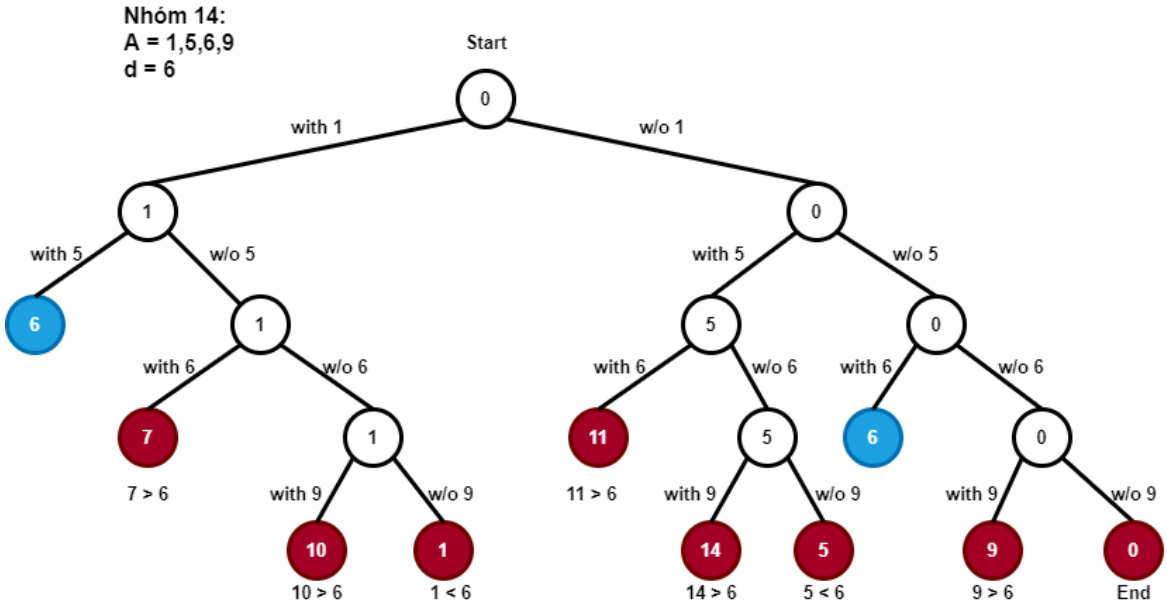
Nhận xét: Nhóm thiếu nhiều trường hợp, sai lầm giống nhóm 3, cần sử dụng tính năng export image.

Nhóm 13:



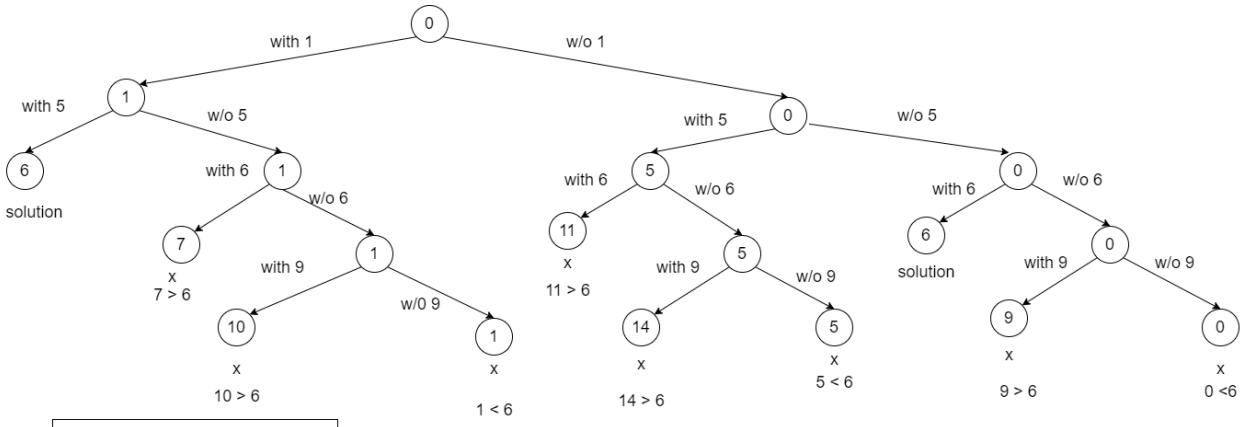
Nhận xét: Nhóm làm đúng.

Nhóm 14:



Nhận xét: Nhóm làm đúng.

Nhóm 15:



-A={1, 5, 6, 9}, d=6

Nhận xét: Nhóm làm đúng.

Nhóm 6: Không nộp bài

Nhóm 10: Không nộp bài

Nhóm 11: Không nộp bài

Tổng kết:

- Nhóm làm đúng và đầy đủ: 2, 13, 14, 15

- Nhóm làm chưa đầy đủ các trường hợp: 3, 4, 5, 7, 8, 9, 12.

- Nhóm không nộp bài: 6, 10, 11