Algorithm Finding pairs (i, j) such that $i \equiv j \mod x$

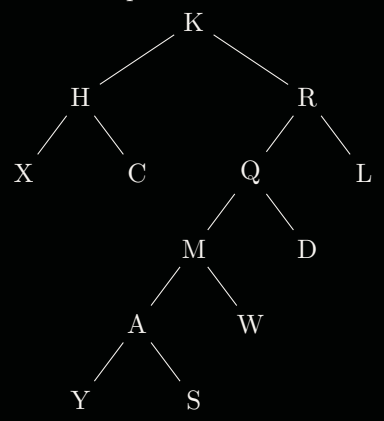
```
function ModPairs(int[] nums, int x)
    n ← length(nums)
    for i ← 1, n − 1 do
        for j ← i + 1, n do
            if i % j = x then
                print("Indices ({i},{j}) with values nums[i], nums[j]")
```

Algorithm Power Set $\mathcal{P}$ (`int[]`)

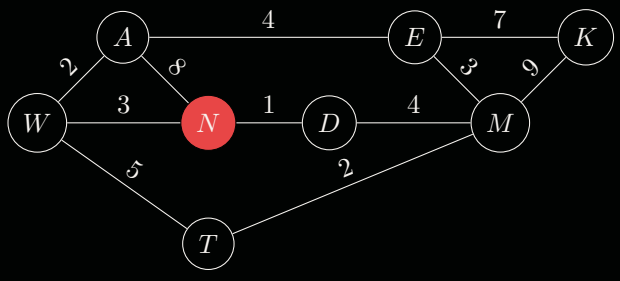
```
function PowerSet(int[]T)
    Queue<int[]> q                                  ▷ declare queue
    q.queue([])                                     ▷ start with ∅
    for all t ∈ T do                                ▷ ∀ t, create new subsets by appending t to all subsets
        while true do                               ▷ iterate through queue until ∅
            int[] subset ← q.deqeue()
            int[] newSubset ← subset.append(t)      ▷ append t to subset

            q.queue(newSubset)                      ▷ queue [subset, t]
            q.queue(subset)                         ▷ requeue subset after
            if subset = [] then
                break                               ▷ stop at ∅
    return q
```

Tree example



Graph (and array) example



Node	distance
D	∞
W	∞
A	∞
M	∞
T	∞
E	∞
K	∞

Algorithm Proposed Critical Section Resolution

```
bool flag [2]                                          ▷ Initially False
int turn;                                              ▷ Initilally 0
do
    flag[i] = True                                     ▷ i == 0 for P_0 and 1 for P_1
    while flag[1-i] do
        if turn == j then
            flag[i] = False
            while turn == j do
                ▷ Do nothing, just wait.
            flag[i] = True
    ▷ Critical Section Code Here
    turn = j;
    flag[i] = False
    ▷ Rest of the Code Here
while True
```