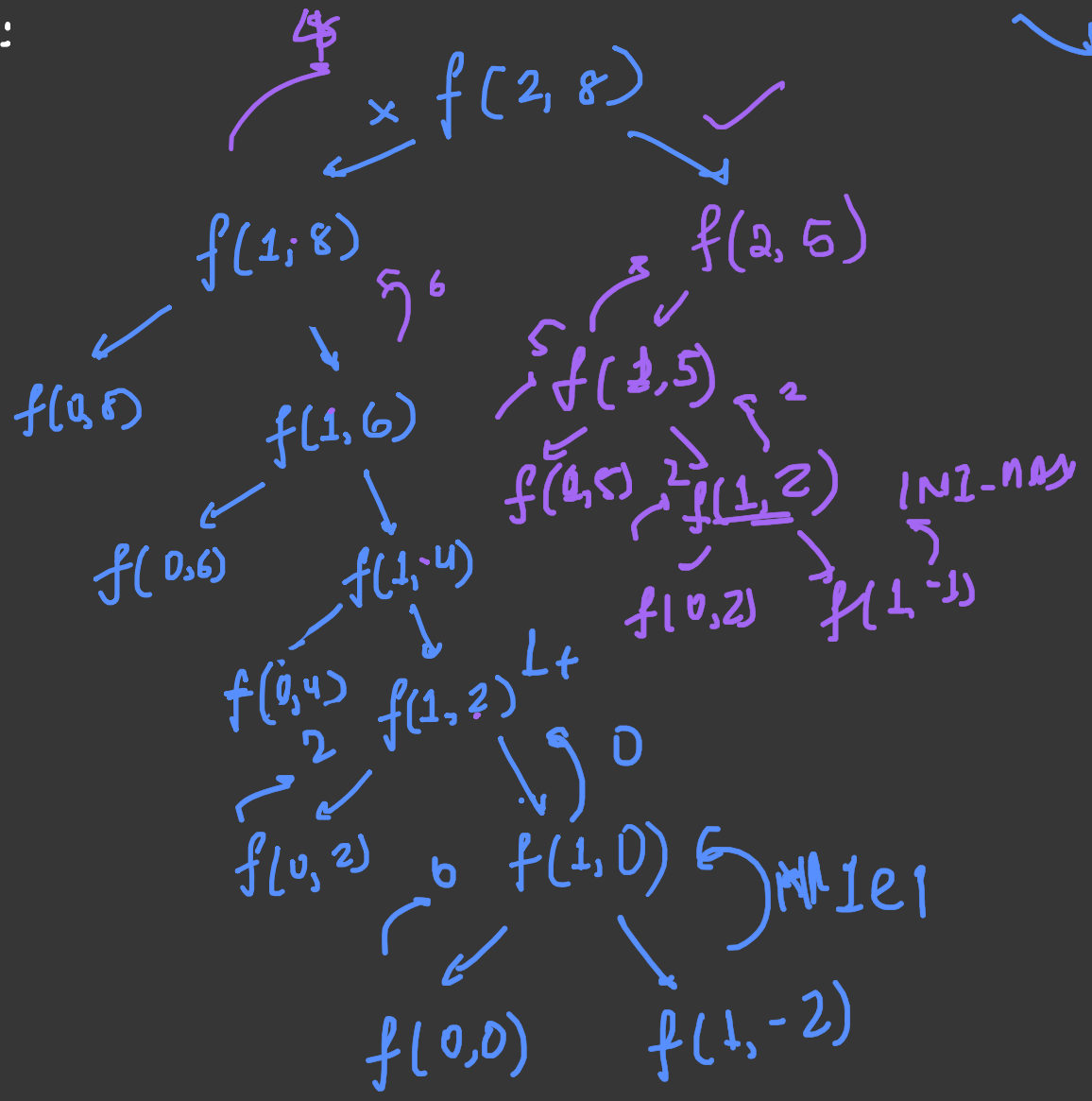
Minimum Coin

arr[i] <= target
pick

Recursion Tree:

arr = {1, 2, 3}

tar = 8



Recursion Relation:

i.e. [ ]

```
if (i == 0){
    if (arr[0] % target == 0) return target/arr[0];
    else return 1e9;
}
```

base case

a[i] > target

```
not pick = fun(i-1, target);

pick = 1e9;
if(arr[i] >= target) pick = 1 + fun(i, target - arr[i]);

return min(pick, not pick);
```

This is different from normal subsequence as we have given infinite value.

T.C. : $O(2^N)$ Exponential

S.C. : $O(N)$ → Auxillary Space

Memonization:

we have 2 changing variable we use dp[n][K]
└ Target

Just add

```
if(dp[i][K]!=-1) return dp[i][K];

{ recursion relation

return dp[i][K] = min(pick, notpick);
```

T.C. : O(N×T)

S.C. : O(N×T) + O(N)
└ auxillary space

Tabulation:

So what is the base case of recursion `if (arr[0] % target) = target/a[0];`

means for every target at arr[0] if it divisible

Base Case

```
for ( T = 0; T <= target; T++)
    if (T % a[0] == 0)
        dp[0][T] = target/a[0];
    else dp[0][T] = 1e9;
```

Nested loop

```
for ( i = 1; i < n; i++)
    for ( T = 0; T <= target; T++)
        notpick = dp[i-1][T];
        pick = 1e9;
        if ( arr[i] <= T) = pick = dp[i-1][arr[i]
                                          T];
```

dp[i][1] = min(pick, notpick);

SpaceOptimise this;

prev,

curr