1235 - Coin Change (IV)

Given n coins, values of them are A_1, A_2 ... A_n respectively, you have to find whether you can pay K using the coins. You can use any coin at most two times.

Input starts with an integer T (≤ 100), denoting the number of test cases.
Each case starts with a line containing two integers n ($1 \le n \le 18$) and K ($1 \le K \le 10^9$). The next line contains n distinct integers denoting the values of the coins. These values will lie in the range $[1, 10^7]$.

For each case, print 'Yes' if you can pay K using the coins, or 'No' if it's not possible.

IDEA:

- ➔ We generate all possible combinations for the left half and right half, and then sort one side, and for each element on the other side, search its corresponding one in the sorted side using binary search.
- ➔ However, We have 3 choices for each item, i.e. take 0, 1, or 2 of it. So, if we try to calculate exactly, generating all possible combination for each side should take $3^{\wedge(n/2)}$ operations which is 3^9 in the worst case. let $m = 3^{\wedge(n/2)}$, then the rest of the thing can be done in O(m log m) using binary search(Meet in the Middle technique).

➔

SampLe Code:

```
int n,hn,k,a[40],sv,su;
vector<int>vv,uu;

void fun1(int pos,int sum)
{
    if(pos==hn){ vv.push_back(sum); return; }

    fun1(pos+1,sum);
    fun1(pos+1,sum+a[pos]);
    fun1(pos+1,sum+a[pos]+a[pos]);
}

void fun2(int pos,int sum)
{
    if(pos==n){ uu.push_back(sum); return; }

    fun2(pos+1,sum);
    fun2(pos+1,sum+a[pos]);
    fun2(pos+1,sum+a[pos]+a[pos]);
}
```

```
bool check(int v)
{
    int lo=0; int hi=su-1;
    while(lo<=hi)
    {
        int md = (lo+hi)/2;
        if(uu[md]==v)return true;
        else if(uu[md]<v)lo=md+1;
        else hi=md-1;
    }
    return false;
}
int main()
{
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++)
    {
        vv.clear(); uu.clear();

        scanf("%d%d",&n,&k);
        for(int i=0; i<n; i++)scanf("%d",&a[i]);
        sort(a,a+n);
        hn = n/2;

        fun1(0,0);
        fun2(hn,0);

        sort(uu.begin(),uu.end());

        sv = vv.size();
        su = uu.size();

        bool flag=false;
        for(int i=0; i<sv; i++)
        {
            int v = vv[i];
            flag = check(k-v);
            if(flag==true)break;
        }

        if(flag==true)printf("Case %d: Yes\n",ks);
        else printf("Case %d: No\n",ks);
    }
    return 0;
}
```