**#include <bits/stdc++.h>**

**using namespace std;**

**#define M 1000000007**

**struct node {**

**int data;**

**node \*left, \*right, \*parent;**

**node(int data) {**

**this->data = data;**

**this->left = nullptr;**

**this->right = nullptr;**

**this->parent = nullptr;**

**}**

**};**

**void insert(node\* root, int data) {**

**if(root->left == nullptr) {**

**node\* n = new node(data);**

**root->left = n;**

**return;**

**}**

**if(root->right == nullptr) {**

**node\* n = new node(data);**

**root->right = n;**

**return;**

**}**

**insert(root->left, data);**

**}**

**void preOrderRecursiveTraversal(node\* root) {**

**if(root != nullptr) {**

**cout<<root->data<<endl;**

**preOrderRecursiveTraversal(root->left);**

**preOrderRecursiveTraversal(root->right);**

**}**

**}**

**void solve() {**

**int n;**

**cin>>n;**

**vector<int>arr(1<<n);**

**for(int i=0;i<(1<<n);i++) {**

**cin>>arr[i];**

**}**

**sort(arr.begin(), arr.end());**

**priority\_queue<int>q;**

**vector<int>ans;**

**if(arr.size() == 1) {**

**return;**

**}**

**if(arr.size() == 2) {**

**cout<<arr[1]<<endl;**

**return;**

**}**

**ans.push\_back(arr[1]);**

**ans.push\_back(arr[2]);**

**q.push(-1\*(arr[1] + arr[2]));**

**for(int i=3;i<(1<<n);i++) {**

**if(q.size() > 0 and arr[i] == -1\*(q.top())) {**

**q.pop();**

**continue;**

**} else {**

**for(int k = 1 ; k < (1<<ans.size()); k++) {**

**int subsetSum = arr[i];**

**for(int j=0;j<ans.size();j++) {**

**if(k & (1<<j)) {**

**subsetSum += ans[j];**

**}**

**}**

**q.push(-1\*subsetSum);**

**}**

**ans.push\_back(arr[i]);**

**if(ans.size() == n) {**

**break;**

**}**

**}**

**cout<<"i: "<<i <<" I am coming"<<endl;**

**}**

**for(auto val: ans) {**

**cout<<val<<" ";**

**}**

**cout<<endl;**

**}**

**int main() {**

**/\* node\* root = new node(6);**

**insert(root, 7);**

**insert(root, 10);**

**insert(root, 12);**

**insert(root, 16);**

**preOrderRecursiveTraversal(root);\*/**

**int t;**

**cin>>t;**

**while(t--) {**

**solve();**

**}**

**return 0;**

**}**