#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

#include<sstream>

#include<stack>

#define MAX

using namespace std;

struct node

{

int weight;

vector<node\*>Child;

};

vector<int>process\_list;

int total\_num, relation, target;

int weight[105];

int comp(node\*a, node\*b)

{

if (a->weight > b->weight)

return 1;

else

return 0;

}

void dfs(node\* current, int weight)

{

weight += current->weight;

if (weight > target)

return;

if (current->Child.size() == 0&&weight==target)

{

cout << process\_list[0];

for (int i = 1; i < process\_list.size(); i++)

cout << " " << process\_list[i];

cout << endl;

return;

}

if (current->Child.size() != 0)

{

sort(current->Child.begin(),current->Child.end() , comp);

for (int i = 0; i < current->Child.size(); i++)

{

process\_list.push\_back(current->Child[i]->weight);

dfs(current->Child[i], weight);

process\_list.pop\_back();

}

}

return;

}

int main()

{

cin >> total\_num >> relation >> target;

for (int i = 0; i < total\_num; i++)

cin >> weight[i];

vector<node\*>nodemap(total\_num);

for (int i = 0; i < total\_num; i++)

{

nodemap[i] = new node;

nodemap[i]->weight = weight[i];

}

for (int i = 0; i < relation; i++)

{

int index, time;

cin >> index >> time;

for (int i = 0; i < time; i++)

{

int temp;

cin >> temp;

nodemap[index]->Child.push\_back(nodemap[temp]);

}

}

process\_list.push\_back(nodemap[0]->weight);

dfs(nodemap[0],0);

return 0;

}