BÀI TOÁN CÁI CÂN

#include <iostream>

#include <vector>

using namespace std;

void printSubset(vector<int>& subset) {

for (int i : subset) {

cout << i << " ";

}

cout << endl;

}

bool isBalanced(vector<int>& weights, int sum, int leftSum, int index, vector<int>& subset1) {

if (leftSum == sum - leftSum) {

printSubset(subset1);

return true;

}

if (index >= weights.size() || leftSum > sum - leftSum) {

return false;

}

subset1.push\_back(weights[index]);

bool include = isBalanced(weights, sum, leftSum + weights[index], index + 1, subset1);

subset1.pop\_back();

bool exclude = isBalanced(weights, sum, leftSum, index + 1, subset1);

return include || exclude;

}

void splitWeights(vector<int>& weights) {

int sum = 0;

for (int weight : weights) {

sum += weight;

}

if (sum % 2 != 0) {

cout << "Khong the chia đeu đuoc.\n";

return;

}

vector<int> subset1;

if (!isBalanced(weights, sum, 0, 0, subset1)) {

cout << "Khong the chia đeu đuoc.\n";

}

}

int main() {

vector<int> weights = {6, 2, 3, 4, 5};

splitWeights(weights);

return 0;

}

BÀI TOÁN GIAO TASK

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct Task {

int reward;

int worker;

Task(int r, int w) : reward(r), worker(w) {}

};

bool compare(Task& a, Task& b) {

return a.reward > b.reward;

}

void distributeTasks(vector<int>& tasks, int m) {

vector<Task> taskList;

int totalReward = 0;

for (int i = 0; i < tasks.size(); ++i) {

taskList.push\_back(Task(tasks[i], 0));

totalReward += tasks[i];

}

sort(taskList.begin(), taskList.end(), compare);

for (int i = 0; i < taskList.size(); ++i) {

taskList[i].worker = i % m;

}

for (int i = 0; i < taskList.size(); ++i) {

cout << "Task " << i + 1 << " co thu lao: " << taskList[i].reward << " cho cong nhan " << taskList[i].worker + 1 << endl;

}

}

int main() {

vector<int> tasks = {5, 8, 4, 7, 2, 6};

int m = 3;

distributeTasks(tasks, m);

return 0;

}  
  
BÀI TOÁN KNAPSACK

#include <iostream>

#include <vector>

using namespace std;

int knapsack(int W, vector<int>& weights, vector<int>& values) {

int n = weights.size();

vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));

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

for (int j = 1; j <= W; ++j) {

if (weights[i - 1] <= j) {

dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - weights[i - 1]] + values[i - 1]);

} else {

dp[i][j] = dp[i - 1][j];

}

}

}

return dp[n][W];

}

int main() {

int W = 10;

vector<int> weights = {2, 3, 4, 5};

vector<int> values = {3, 4, 11, 6};

cout << "Tong gia tri lon nhat co the: " << knapsack(W, weights, values) << endl;

return 0;

}