**Algorithm LAB 6**: Knapsack Problem

**Name**  : Muhle Musa Khumalo

**StudentID** : 410921335

**Date**  : April 29, 2022

Solution:

#include <iostream>

#include <string>

#include <vector>

using namespace std;

int main()

{

int T;

cin >> T;

while ( T-- )

{

// Ferry

int ferry;

cin >> ferry;

ferry \*= 100;

int car;

vector<int> cars(1, 0);

// sum[i] stores the total length of cars 1 to i.

vector<int> sum(1, 0);

while (cin >> car, car > 0)

{

cars.push\_back(car);

sum.push\_back(sum.back() + car);

}

// dp[i][j] is true if cars 1 to i could be loaded for length j on right.

vector<vector<bool> > dp(cars.size(), vector<bool>(ferry + 1, false));

vector<vector<string> > lane(dp.size(), vector<string>(dp[0].size(), ""));

pair<int, int> memo = make\_pair(0, 0);

dp[0][0] = true;

for (int i = 1; i < (int)cars.size(); ++i)

for (int j = 0; j <= ferry; ++j)

{

// If cars 1..i-1 were successfully loaded for length j on right.

if (dp[i - 1][j])

{

// Try to load car i on right.

if (ferry - j >= cars[i])

{

dp[i][j + cars[i]] = true;

lane[i][j + cars[i]] = "starboard" ;

memo = make\_pair(i, j + cars[i]);

}

// Try to load car i on left.

if (ferry - (sum[i - 1] - j) >= cars[i])

{

dp[i][j] = true;

lane[i][j] = "port";

memo = make\_pair(i, j);

}

}

}

cout << memo.first << endl;

string print;

while (lane[memo.first][memo.second] != "")

{

print = lane[memo.first][memo.second] + "\n" + print;

memo.second -= lane[memo.first][memo.second] == "starboard"?

cars[memo.first] : 0;

--memo.first;

}

cout << print;

if (T)

cout << endl;

}

return 0;

}

Result on OJ:

Text

Description automatically generated

Complexity analysis

**Time Complexity: O(N\*W)** as redundant loading of cars is avoided

Runtime = T\*(N+(N\*W)+W) where T is number of test cases, N is number of cars and W is the number of cars that fit within the ferry length. Others are O(1)

**Auxiliary Space**: **O(N).**The use of vector and pair data structure for storing cars and relevant data. At most space use is O(2\*N) for pairing data in vector and pairs. That is class of O(N)

Sum = O(N)

Car = O(N)

Dp = O(2\*N)

Lane = O(2\*N)

Memo = O(2\*N)

Others are O(1)