**I. Phương pháp tham lam**

**1.Mô tả**

Phương pháp tham lam đưa ra quyết định dựa ngay vào thông tin đang có, và trong tương lai sẽ không xem xét lại tác động của các quyết định trong quá khứ.

Chính vì thế các thuật toán dạng này rất dễ đề xuất, và thông thường chúng không đòi hỏi nhiều thời gian tính.

Tuy nhiên, các thuật toán dạng này thường không cho kết quả tối ưu.

**2.Ý tưởng**

Xuất phát từ lời giải rỗng, thuật toán xây dựng lời giải của bài toán theo từng bước, ở mỗi bước sẽ chọn một phần tử từ tập ứng cử viên và bổ sung vào lời giải hiện có.

Hàm Solution(S) nhận biết tính chấp nhận được của lời giải **s.**

Hàm Select(C) chọn **từ** tập **c** ứng cử viên có triển vọng nhất để bổ sung vào lời giải hiện có.

Hàm Feasible(S+x) kiểm tra tính chấp nhận được của lời giải bộ phận s+x.

**3.Lược đồ chung**

procedure **Greedy;**

(\* Giả sử c là tập các ứng cử viên \*) begin

S: = 0; (\* s lời giải xây dựng theo thuật toán \*) while (C \* 0) and not Solution(S) do begin

X <- Select(C);

C:= c \ x;

if Feasible (Sux) then S:=Sux; end;

if **Solution(S)** then return **S;** end;

II.Bài toán cái túi sử dụng phương pháp tham lam

1.Bài toán

(Knapsack Problem)

Có n đồ vật đồ vật i có trọng lượng Wị và giá trị Cj, i = 1,2, n.

Tìm cách chất các đồ vật này vào cái túi có dung lượng là b sao cho tổng trọng lượng của các đồ vật được chất vào túi là không quá b, đồng thời tổng giá trị của chúng là lớn nhất.

**2.Ý tưởng**

### Ý tưởng (ít tham lam): Đồ vật có đơn giá lớn (nhất) còn lại được lấy trước (nếu có thề).

### Chi tiết:

Sắp xếp các đồ vật theo thứ tự không tăng của giá trị một đơn vị trọng lượng (cl/wl)

Chọn đồ vật từ đầu đến cuối ...

**3.Thuật toán**

Sắp xếp theo giá tri/trọng lương

/\*! Sắp xếp đồ vật theo thứ tự giảm dần của giá trị \*/

void SortObj(int left, int right)

{

float pivot = ob[(left + right) / 2].RealValue;

int i = left;

int j = right;

while (i <= j)

{

while (ob[i].RealValue > pivot)

i++;

while (ob[j].RealValue < pivot)

j--;

if (i <= j)

{

Obj tmp = ob[i];

ob[i] = ob[j];

ob[j] = tmp;

i++;

j--;

}

}

if (left < j)

SortObj(left, j);

if (i < right)

SortObj(i, right);

}

Hàm GreedyPick

void GreedyPick(int W)

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

if (W > ob[i].Weight)

ob[i].SoLuong = 1;

W -= ob[i].Weight;

total += ob[i].SoLuong;

totalValue += ob[i].Value \* ob[i].SoLuong;

totalWeight += ob[i].SoLuong \* ob[i].Weight;

**4. Cài đặt**

#include <iostream>

#include <fstream>

using namespace std;

#define MAX 100

struct Obj{

int id; //luu id cua vat

int Weight, Value;

float RealValue;

int SoLuong; // do vat duoc chon

};

int W[MAX + 1], V[MAX + 1];

int n; // so luong do vat

int w; // khoi luong cua balo

int total = 0, totalValue = 0, totalWeight = 0;

Obj ob[MAX + 1];

void Input(){

ifstream infile;

infile.open("input3.txt");

infile >> n >> w;

cout << n << " " << w << "\n";

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

{

infile >> ob[i].Weight >> ob[i].Value;// trong luong va gia tri cua moi do vat

cout << "#" << i << ": " << ob[i].Weight << " " << ob[i].Value << "\n";

}

infile.close();

}

//Tính gia tri / khoi luongcua d vat

void CalculateRealValue()

{

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

{

ob[i].id = i; // set id

ob[i].RealValue = (float)ob[i].Value / (float)ob[i].Weight;

}

}

/\*Sap xep giam dan\*/

void SortObj(int left, int right)

{

float pivot = ob[(left + right) / 2].RealValue;

int i = left;

int j = right;

while (i <= j)

{

while (ob[i].RealValue > pivot)

i++;

while (ob[j].RealValue < pivot)

j--;

if (i <= j)

{

Obj tmp = ob[i];

ob[i] = ob[j];

ob[j] = tmp;

i++;

j--;

}

}

if (left < j)

SortObj(left, j);

if (i < right)

SortObj(i, right);

}

void GreedyPick(int W)

{

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

{

//cout<<ob[i].id<<" ~ "<<ob[i].Weight<<" ~ "<<ob[i].Value<<" ~ "<<ob[i].RealValue<<endl; // debug

if (W >= ob[i].Weight)

{

/\* uncomment loai 2 \*/

//ob[i].SoLuong = W / ob[i].Weight;

//W -= ob[i].SoLuong \* ob[i].Weight;

/\* uncomment v?i bàiloai 1 \*/

ob[i].SoLuong = 1;

W -= ob[i].Weight;

total += ob[i].SoLuong;

totalValue += ob[i].Value \* ob[i].SoLuong;

totalWeight += ob[i].SoLuong \* ob[i].Weight;

}

}

}

void Print()

{

cout << "\nTong so vat da lay: " << total;

cout << "\nGia tri lon nhat co the lay: " << totalValue;

cout << "\nKhoi luong da su dung: " << totalWeight;

cout << "\nCac vat da lay [stt (so luong)] : ";

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

{

if (ob[i].SoLuong > 0)

{

cout << ob[i].id << " (" << ob[i].SoLuong << ") "; // chon vat i - so luong ob[i].SoLuong

}

}

}

int main()

{

Input();

CalculateRealValue();

SortObj(1, n);

GreedyPick(w);

Print();

return 0;

}

**5.Thực hiện chạy các bước**

Bộ dữ liệu 1:



Các bước thực hiện:

Sau khi sắp xếp theo v/m

Sắp xếp



Hàm





Bộ dữ liệu 2:



Các bước thực hiện

Sắp xếp



Hàm Greedy





Bộ dữ liệu 3:



Các bước thực hiện

Sắp xếp



Hàm Greedy





Bộ dữ liệu 4:



Các bước thực hiện

Sắp xếp



Hàm Greedy



Bộ dữ liệu 5:



Các bước thực hiện

Sắp xếp



Hàm Greedy





**6. Đánh giá độ phức tạp của thuật toán**

Độ phức tạp thuật toán O(n)