演算法作業\_0-1knapsack

楊子右 電機三甲 E24066250

編譯環境:

Visual Studio 2017,Linux gcc version 7.5.0

Input=”input.txt”

Output=”output.txt”

* Merge Sort

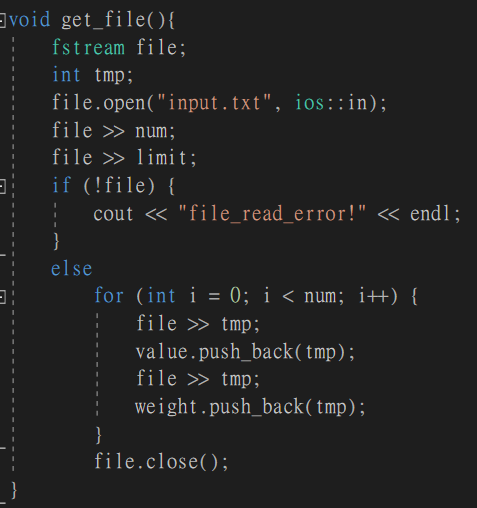
演算法以及實現想法:

遍歷所有物品以及所有限制重量內的重量決定最佳解，每次決定放入或不放入，挑最好的結果進行。在子問題為最佳解的組合情況下再進去判斷，之後都會得到最佳解，最後得到最佳總問題的解答。

* 二元搜尋樹之實作與其功能實現

1. get\_file()

Code:

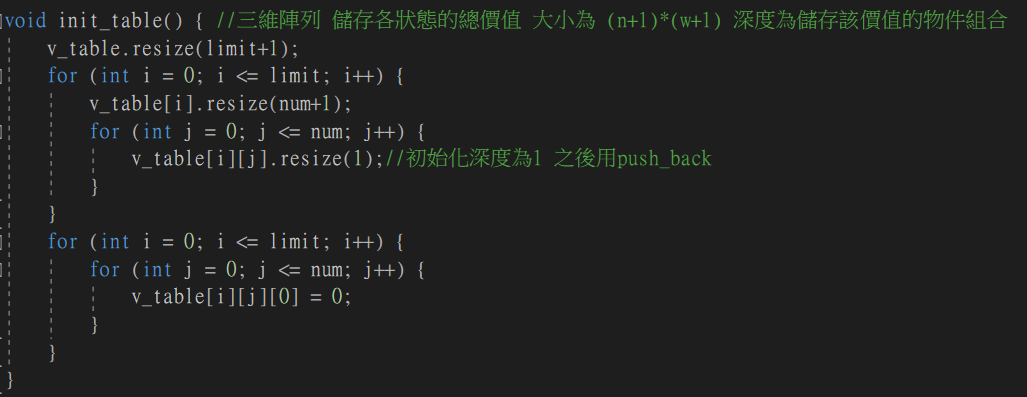


想法:

讀入測資

1. init\_table()

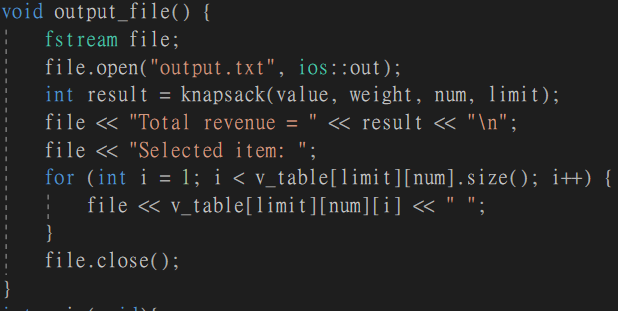
Code:

 想法:

初始化一個皆為0之三維陣列，高為重量限制，寬以及深度為物品個數。在這陣列上進行動態規劃。

1. output\_table()

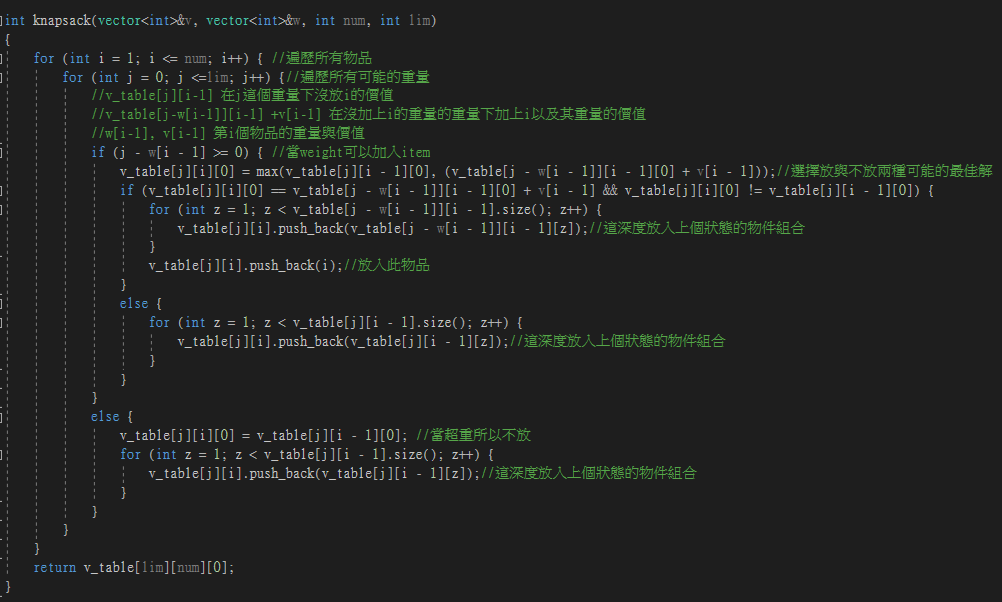
Code:



想法:

輸出結果

1. knapsack(vector<int>&v, vector<int>&w, int num, int lim)

Code:

想法:

遍歷所有物件以及重量 在 (j-w[i-1]>=0) 也就是背包能夠承受第i個物件的重量時決定放或不放。不論放或不放，在該格的深度都要放入該狀態的物件集合(for (int z = 1; z < v\_table[j - w[i - 1]][i - 1].size(); z++))。若超重則不放，最後回傳v\_table[lim][num][0] 所有深度的第一個(v\_table[j][i][0])為該狀態下的最佳解。

Vector v, w 為按照順序儲存所有物件之價值以及重量。