CPP1002 Coding Assignment 4

背包問題

電子檔 2011/04/18 09:00 前上傳至 moodle
書面檔 2011/04/18 09:15 前於課堂繳交

問題描述

假設現有一網路商家出國批貨，現已知此網路商家僅攜帶一行李箱且其承載重量上限為W，因此其在批貨的時候，需要依據商品的重量與利潤考慮要帶哪幾項商品回來以獲得最大的利潤。若目前正在考慮購買的商品共有 n 項，其重量分別為$w_0, w_1, ..., w_{n-1}$，而每樣商品所預估可以獲得的利潤分別為(此利潤是指扣除商品成本後的價值) $p_0, p_1, ..., p_{n-1}$。本題旨在替該網路商家計算最佳的商品選購決策（亦即應該購買那幾項商品），以在有限的行李箱重量限制內，獲取最大的總體利潤。

此問題為傳統「背包問題」(knapsack problem)的直接應用，背包問題的應用極廣，譬如物流業中的貨物裝箱裝櫃作業，或是人造衛星中科學實驗器材的選定等等。雖然該問題之描述方式淺顯易懂，然其已被證明為一個理論上極難求解的問題。簡而言之，到目前為止仍無人能針對該問題提出有效率之解法，隨著 n 值變大，求解其最佳解所會耗費的計算時間將成長十分快速（舉例來說，n=30 可能已經無法用一般的電腦在幾小時內求出最佳解）。

為了讓同學體驗一下這種「看似簡單、實則很難」的求解經驗，本次作業擬要求同學以 C++ 程式使用(1)暴力法(Brute Force)；(2)貪婪法(Greedy Algorithm)來實作此問題：

(1) **暴力法：**即為「窮舉法」(Total Enumeration)，也就是列舉出此問題所有可能的解，計算各個解的重量與利潤後，找出符合限制的最佳解。在背包問題中，暴力法即為列出所有放入背包中商品的可能組合，判斷這些組合是否符合背包重量的限制(亦即總重量是否小於或等於W)，接著計算各個商品組合之總利潤，最後找出符合總重量限制之利潤最高的組合。

由上述可知，由於每一個商品皆有「放入」與「不放入」背包兩種可能，因此可以推知共有2^n-1種組合(排除全部不放入背包的情形)。我們可以利用一個$1\times n$大小的 0-1 x 陣列來表示選取商品的組合，如下圖(一)所示。假設現在有 7 樣商品$i \in \{0,1,2,3,4,5,6\}$(為方便起見，本作業的商品編號從 0 開始算)，以$x_i \in \{0,1\}$表示選取第i樣商品放入背包($x_i=1$)中與否($x_i=0$)。依此可推出，圖(一)表示將商品 1、3、4 及 6 放入背包中。

x_0	x_1	x_2	x_3	x_4	x_5	x_6
0	1	0	1	1	0	1

圖(一)

在窮舉的過程中，我們若將數字 1 至2^n-1利用二進位的方式表達，便可以得到所有商品的組合方式，以下將以 n=3，即共有 3 樣商品的情形，簡略說明二進位與x陣列之關係。當數字以二進位表達時，則十進位的 2 需要以 10 表示，3 以 11 表示，4 以 100 表示；依此類推，數字 1 至 7(亦即2^3-1)以二進位方式可表達如下

數字	x_0	x_1	x_2
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

圖(二)

其中數字 1 表示將商品 2 放入背包中；數字 2 表示將商品 1 與 2 放入背包中；而數字 1,…,7(或 1,…,2^n-1)即可透過二進位的表示法來代表 3(或 n)種商品的所有組合方式，若組合方式已知便可以輕易的求出該商品組合方式所對應的總重量($\sum_{i=0}^{n-1} w_i x_i$)以及總利潤($\sum_{i=0}^{n-1} p_i x_i$)，從而找出在重量限制範圍內($\sum_{i=0}^{n-1} w_i x_i \le W$)，獲得最高利潤的組合($x^*$)。

(2) 貪婪法：上面我們所說明的暴力法，是在所有可能解中找出最好的結果，因此必能找到最佳解。然而，隨著商品的項目數 n 越大，所有可能的組合個數將呈指數型態(2^n)增加，因此暴力法的求解時間亦將隨著商品項目增加而快速增加，致使無法在合理的時間內求解。

在實際應用上，由於求取最佳解可能耗時過久，因此實務上經常退而求其次，僅希望在合理的時間內可計算得一個品質還不錯的近似解。此時，「短視近利」的貪婪演算法(greedy algorithm)通常會被用來求出一組「好」的解。貪婪法的主要精神乃在其求解過程中以符合直覺的有系統方式來朝最佳解之目標邁進，由於其過程簡易且有效率，所得之解與最佳解可能不會差距太遠，因此通常被使用於計算複雜問題的初始解。

接下來我們將說明用以求解此背包問題的一種貪婪法：由於背包問題之目標在於找到利潤最大且符合重量限制的商品組合，直覺上我們應該會傾向選取利潤愈大但重量愈小的商品來裝背包，以數學的術語來說，也就是我們可根據各商品利潤與重量的「利重比」p_i / w_i，從大至小排序，依序一一選取商品裝背包，直至背包限重將被違反卻又符合限制為止。這樣的作法即是以「單位重量所擁有的利潤」為唯一的商品選取條件，可同時兼顧利潤愈大但重量愈小的兩種直覺。此外，此方法之過程中皆以「貪婪」(即從大至小)的方式選取商品，然而僅考慮此一利重比(即「短視近利」)的方式並無法保證必定可選出最佳商品組合。

本作業希望比較暴力法與貪婪法的優缺點，因此須分別計算出兩種方法的求解結果以及求解時間，讓同學藉由實作來驗證以下事項：

- 當商品的數目不多時，可以在短時間內利用暴力法求得最佳的結果，但是當商品數目增加時，則暴力法將耗時甚久
- 貪婪法皆可快速求解，但其求解結果將不一定為最佳解

程式要求及作法

本作業大概可以分成以下幾個部分：

一、**讀輸入檔並建構所需要之陣列**：以一函式 initialize()實作之

Input: filename

Output: n, W, w[n], p[n], Rpw[n], x[n], XB[n], XG[n], Rorder[n]

其中 fileName 為使用者所輸入之檔名，本作業提供 n4.txt、n16.txt 以及 n32.txt 三個測試輸入檔，以下以 n4.txt 為例說明各檔案所包含的資料：

```
4        //表示有 4 項商品, n=4
15       //表示重量的上限值為 15, W=15
5 15     // w[0]=5；p[0]=15：表示商品 1 的重量為 5，價值為 15
2 12     // w[1]=2；p[1]=12：表示商品 2 的重量為 2，價值為 12
7 17     // w[2]=7；p[2]=17：表示商品 3 的重量為 7，價值為 17
9 19     // w[3]=9；p[3]=19：表示商品 4 的重量為 9，價值為 19
```

在此函式中，必須根據讀入的 n 值透過動態配置記憶體的方式宣告大小適合的陣列：

double w[i]：i=0,…,n-1，用以存取重量 w_i

double p[i]：i=0,…,n-1，用以存取利潤 p_i

double Rpw[i]：i=0,…,n-1，用以存取「利重比」 p_i / w_i

bool x[i]：i=0,…,n-1，用以存取某組商品選取方式

bool XB[i]：i=0,…,n-1，用以存取暴力法的最佳商品選取方式

bool XG[i]：i=0,…,n-1，用以存取貪婪法的最佳商品選取方式

int Rorder[i]：i=0,…,n-1，用以存取商品依其利重比從大至小排序後的商品編號

舉例來說，Rorder[0]=3 代表利重比最大的為商品 3（亦即，貪婪法第 1 個會選取的商品為商品 3），而 Rorder[5]=4 代表商品 4 之利重比排名為第 6 大（亦即，貪婪法第 6 個會選取的商品為商品 4）

本部分困難之處：此處我們要求同學將全部的初始化部分以一個函式來處理，因此必定會遇到如何用pointer 讓函式可以一次回傳多個值(包括array)及如何開檔讀檔等諸多問題。老師建議同學們可以先在main()裡面處理好這個部分，再試圖將這個部分全部包成一個initialize()的函式來處理。此外，助教亦會於實習課教授以第三個作業的map檔為例之範例程式供大家參考（請下載 http://ilin.iim.ncku.edu.tw/ilin/course/CPP1002/example/init_examples.rar）。

二、**印出求解時間**：由於題目要求印出分別印出暴力法與貪婪法求解的時間，因此需使用 C++內建的 function 來處理，必須 include <ctime> 標題檔。C++中可用來計時的函式不只一種，以下僅說明其中一種方法給同學作參考：

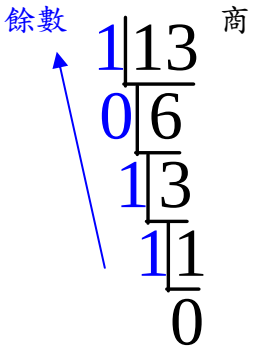
```
#include <ctime> //需要加入的標題檔
time_t t_start, t_end; //變數宣告
double duration=0.; //記錄時間長度
```

```
t_start = time(NULL); //記錄開始的時間
    //欲被量測時間之程式區塊
t_end = time(NULL); //記錄結束的時間

duration = difftime(t_end, t_start); //計算經過的時間，單位：秒
```

另外，亦可用clock_t的方式來記錄時間（單位為毫秒），詳情可參考課堂講義 http://ilin.iim.ncku.edu.tw/ilin/course/CPP1002/example/get_duration.cpp。

三、**暴力法**：主要流程為以n個0-1來表示商品的選取組合，這裡可以利用C++內建之pow(2,n)函式計算2^n之值（記得必須 include <math.h> 標題檔），要注意的是當 n 值很大時（譬如大於 32）2^n可能會產生溢位(overflow)，此時可用可存取更大整數的 unsigned int 或long long int來宣告用以存取2^n-1的變數(假設為a)，接著藉由變動不同的a值(a=1,…,2^n-1)而將其二進位的表達方式存入x陣列中。其中，從a轉換成x的過程可以使用以2為除數的連續除法，將得到的餘數(非0即1)依序一一存入x[n-1],x[n-2],…x[1],x[0]。以n=4為例，a將以每回合增加1的方式自a=1開始持續變動至a=$2^4-1=15$。圖(三)為a=13的轉換例子，透過短除法，13/2=6…1（即 x[3]=1）；6/2=3…0（即 x[2]=0）；3/2=1…1（即 x[1]=1）；1/2=0…1（即 x[0]=1；當除至商=0時即可停止此連續除法）。在得到某個a值所對應的x後，透過陣列p,w可計算該組合x所產生的總重量($\sum_{i=0}^{n-1} w[i]x[i]$)與總利潤($\sum_{i=0}^{n-1} p[i]x[i]$)；針對符合重量範圍內($\sum_{i=0}^{n-1} w[i]x[i] \le W$)的組合x，若其總利潤高於目前已存取的最高利潤組合，則將此目前最佳的組合x記錄於XB陣列中（亦即XB=x），反之則繼續用下個a來計算x。當所有的a都輪過一次後，XB即記錄整體而言最佳的商品組合方式。



圖(三)

四、**貪婪法**：此部分主要分別計算各商品之利重比Rpw[i]=p[i]/w[i],i=0,…,n-1，接著以一函式Get_Rorder()將Rpw[]進行從大至小的排序(可用任何排序方法)，並回傳Rorder[]至main()，再依Rorder[]一一選取商品直至無法再選取為止。最後將剛剛的選取方式存入XG陣列中，此部分主要須小心使用迴圈與判斷式使商品組合之重量和在允許範圍內。

HW 4, 電子檔 2011/04/18 09:00 前上傳至 moodle
書面檔 2011/04/18 09:15 前於課堂繳交
http://ilin.iim.ncku.edu.tw/ilin/course/CPP1002/

本部分困難之處：函式 Get_Rorder()可能會讓同學花較多時間處理，因為這個部分牽涉到 array 於 function 的傳入與回傳，且該函式主要功能在於將 Rpw[]排序。在此老師建議同學們可以先將這個部分寫成一個程式來測試，先假設你有一組 Rpw[]，思考如何才能將其排序並同時將其排序存於 Rorder[]陣列。確認這個部分可以成功執行之後，再將其 copy/paste 至本程式內即可。排序部分可用最常見的 bubble sort 或 insertion sort 等等排序方式，其排序原理很容易理解，網路上隨便找都有很多資訊可參考。

綜合流程說明與輸出結果

分別測試檔案"n4.txt"、"n16.txt"以及"n32.txt"，以"n4.txt"為例，讀取檔案之後，建立相關的陣列以及儲存商品相關的資料(第一部分)，接著進行暴力法(第三部分)以及貪婪法的流程(第四部分)，其中必須分別計算兩種方法的求解時間(第二部分)，並且印出類似以下的結果：

```
------- input file: n4.txt ------
BF: Max profit: 44
    Selected items: 0 1 2
    Time taken: 0ms (0s)
GA: Max profit: 44
    Selected items: 1 0 2
    Time taken: 0ms (0s)
```

其他注意事項：

- 在時間的計算方法中，如果是依秒計，在 n 小的時候，可能都是近乎 0 秒；在 n=16 時還不一定可以跑超過 0.1 秒，所以同學在測試小 case 時，可能會顯示為 0 秒。或許大家在測試小 case 時，可用毫秒為單位來計算演算法耗費之時間長度。
- 雖然我們僅提供 3 組測試檔，助教在評分時可能會再用更大的例子來檢查同學的程式，請同學可自行設計更多例題來測試其程式之正確性。

***作業繳交應注意事項**

1. 作業需要繳交電子檔以及書面(列印程式檔及程式結果)
2. 電子檔請於作業繳交截止時間以前上傳至http://moodle.ncku.edu.tw
   2.1 請同學先建立一個**資料夾**，資料夾名稱為“**組別_學號_hw4**”，例如組別為第 4 組，學號為 hxxxx，則資料夾名稱則為 **g4_hxxxx_hw4**
   2.2 將程式檔案名稱存為“**hw4.cpp**”，並將之存於上述設立之**組別_學號_hw4** 資料夾中
   2.3 最後將整個**組別_學號_hw4** 資料夾壓縮成 zip 檔(**組別_學號_hw4.zip**)，再上傳至 moodle (！注意！：**請勿**將 cpp 檔 copy/paste 至 word 檔而上傳之)
3. 書面作業請於 **2011/04/18 上課 5 分鐘內(09:15 前)**繳交至講台，其中需要註明程式是否能被編譯與執行、撰寫人、程式之目的、如何編譯及執行等資料(詳見 http://ilin.iim.ncku.edu.tw/ilin/course/CPP1002/programming.html)。