**Algorithm Report**

**B09901066 謝承修**

我用了vector紀錄所有輸入，並且開了兩個長度為w的一維vector (named as old and cost)，利用iteration的方式紀錄cost(W, n)的最小cost，滾動式的交替紀錄最佳解，時間複雜度為O(nW)，時間複雜度為O(W)。

基本上迭代的方式為以下式子：

**cost[j] = min(cost[j - 1], old[j - w[i]] + e[i] \* abs(x[i] - (j - w[i])));**

然後初始化的方式是將填不進w[1]的格子設為無窮大，其餘依次遞減直至0。

然後每次執行完都會把cost的值丟給old，持續跑完整個W。

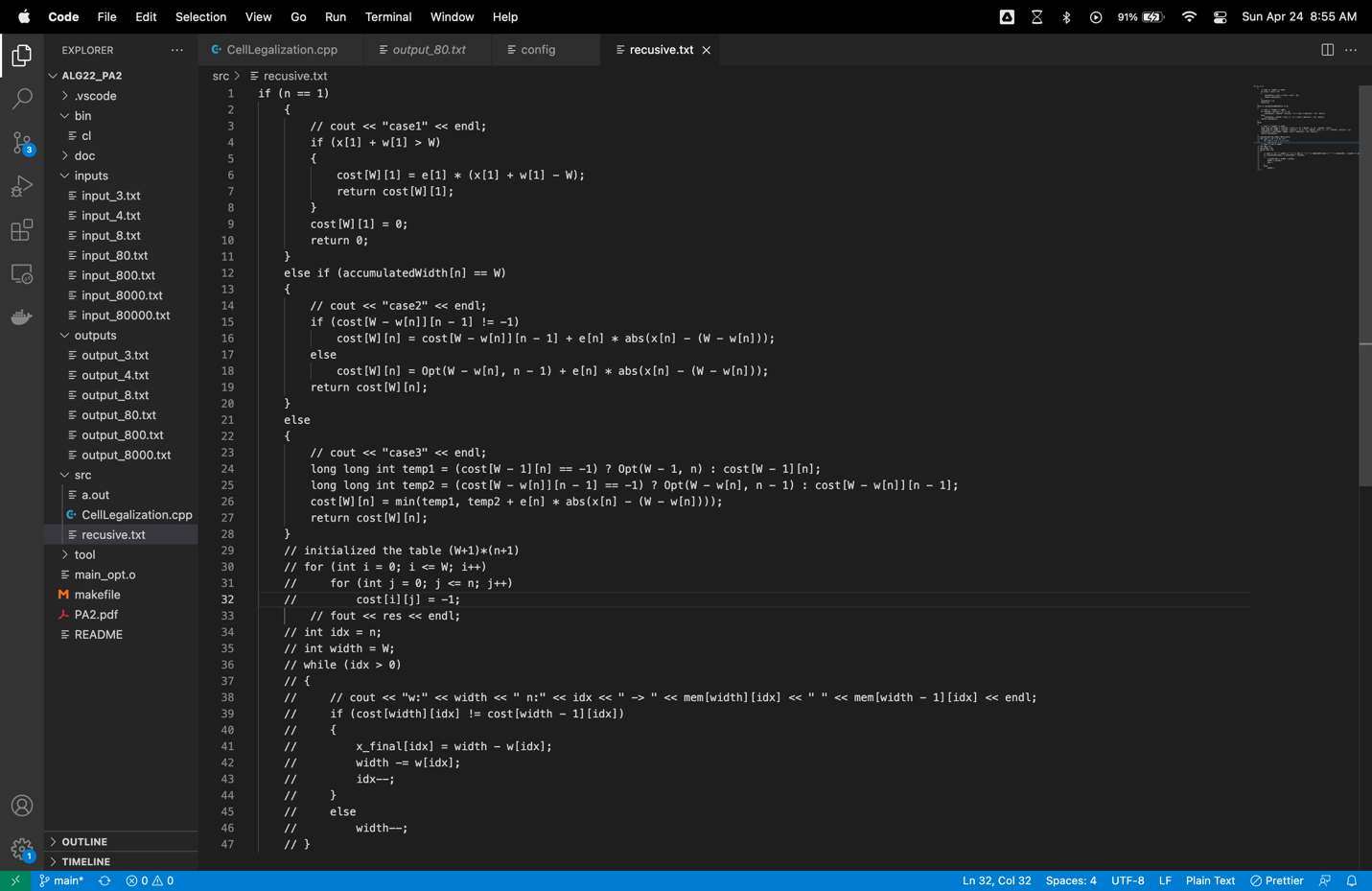
最後需要回推出每個x的位置，方式也是透過建立兩個vector，由上述式子移項反推前一層optimal cost (如以下式子)，然後依序紀錄最後一個cell的最終位置。雖然因為只由一維的資料反推會比較繁雜，但是卻能有效降低空間複雜度。

**curCost = e[idx] \* abs(x[idx] - (j - w[idx])); old1[j - w[idx]] = cost1[j] - curCost;**

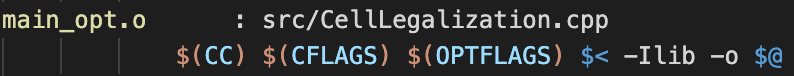
另外我發現index應該要前後一致才對，不然很容易出bug，我認為從1會比較符合直覺，所以所有的vector前面都有先塞進一個零。

**參考資料 ：[背包问题（0-1背包、完全背包问题）（动规、滚动数组）](https://fanxinglanyu.blog.csdn.net/article/details/104461218?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~Rate-1.pc_relevant_antiscanv2&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~Rate-1.pc_relevant_antiscanv2&utm_relevant_index=2&fbclid=IwAR3zbyBKlZEUc78ioy9WiS3gGD4-RxMWmkOa0xNdhZEXkpht4Ej7KZUQ8qI)**

附圖為recusive的寫法，寫完之後發現input 8000的都跑不出來，所以改寫為iteration+滾動數組，空間複雜度由O(nW)降為O(W)，時間複雜度則維持不變。

****

**補交修改之處**

****

