程式設計 期中專題

組員 : B07705010黃煒勛、B07705011黃靖翔、B07705013黃心、B07705045呂皓恩

系統設計

* **Header files :**

<bits/stdc++.h> - 引入所有標準函式庫

pragma optimize("Ofast") - 最佳化編譯器

* **Max parameters :**

定義const int max\_n, max\_m, max\_point, max\_k作為各項參數最大值

* **Time limit :**

設定時間限制為980ms，在時間超出此範圍便輸出當前得到之解，避免超時而無結果輸出

* **Definition :**

定義各項變數、function header

void SimulatedAnnealing : 主要演算法

void SearchForNewSolution : 決定如何產生新路線、更新新路線

void UpdateSolution : 根據threat回傳結果決定是否採用新路線

int PointsOfRoute : 計算點數 i.e.轉折次數

double LenOfRoute : 計算路線總長度

double threat : 計算威脅

* **Main Function :**

設定time作為random seed、輸入資料、呼叫SimulatedAnnealing執行演算法、輸出結果

演算法

**模擬退火法之應用(simulated annealing) :**

此為一種機率演算法，用來在一定時間內尋找搜尋空間內的近似最優解，因這次題目中所有路線的可能性過多，無法一一窮舉並計算其威脅

**初始化 :**

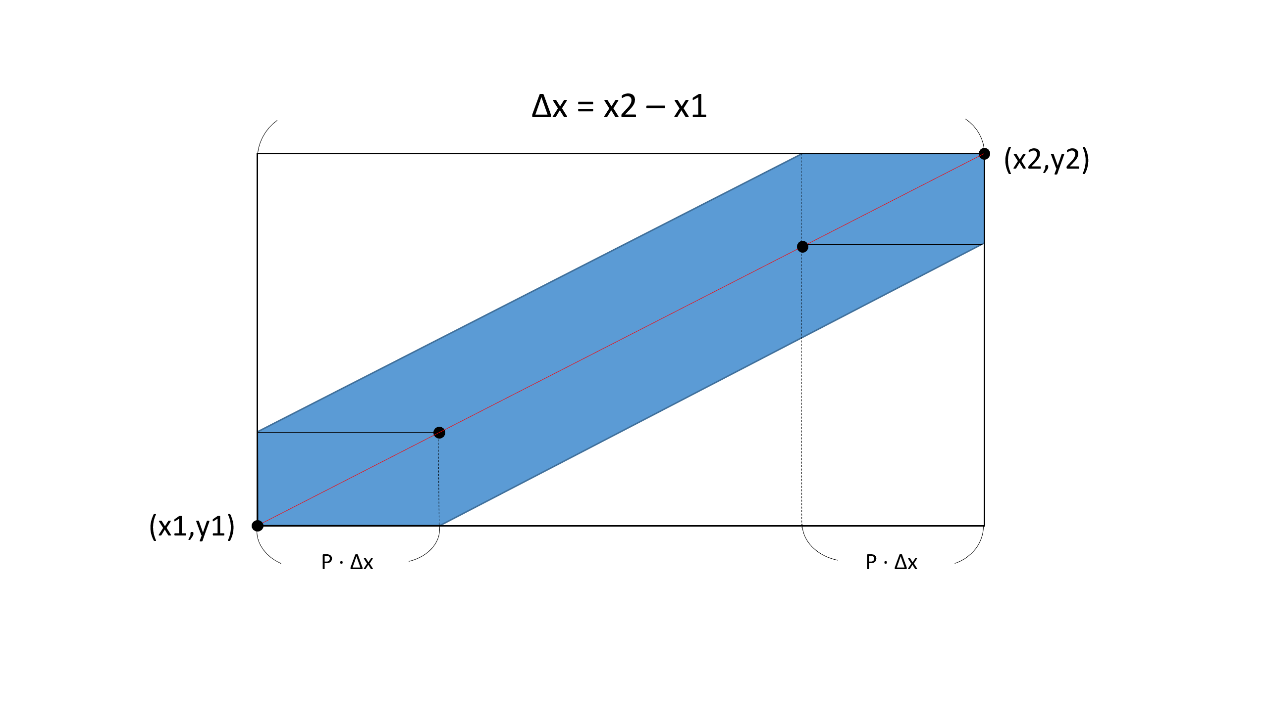
生成初始解(起點到終點，無轉折)，xy座標矩陣中其餘值為-1(表示未走過)

**過程 :**

1. 產生新解，本次使用random挑選增加一點、減少一點、改變一點其中一方法，在其中一(用random挑選)路段附近進行變化，便於後續計算

**取點方法(增加一點、改變一點) :**

在圖中所示範圍中隨機取一點，避免隨機範圍過大



1. 計算threat函數差，並判斷是否被接受。

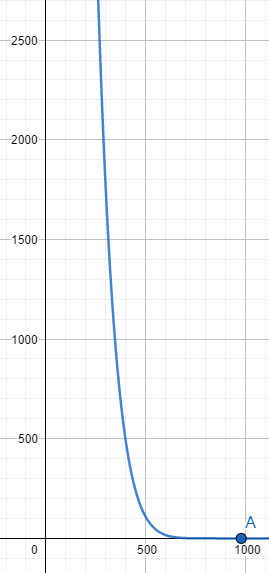
**判斷方式 :**

若威脅較小，直接接受為新解

威脅較大則以一遞減函數機率性判斷是否接受

rand() < pow((1.0 \* (clock()-timelim) / timelim), power) \* RAND\_MAX

此函數在clock()=0時函數值為RAND\_MAX，clock()≠0時則隨clock()增長急速下降，至time limit時函數值為0，代表隨執行次數增加，對較差解的接受度下降，越容易找出較佳解



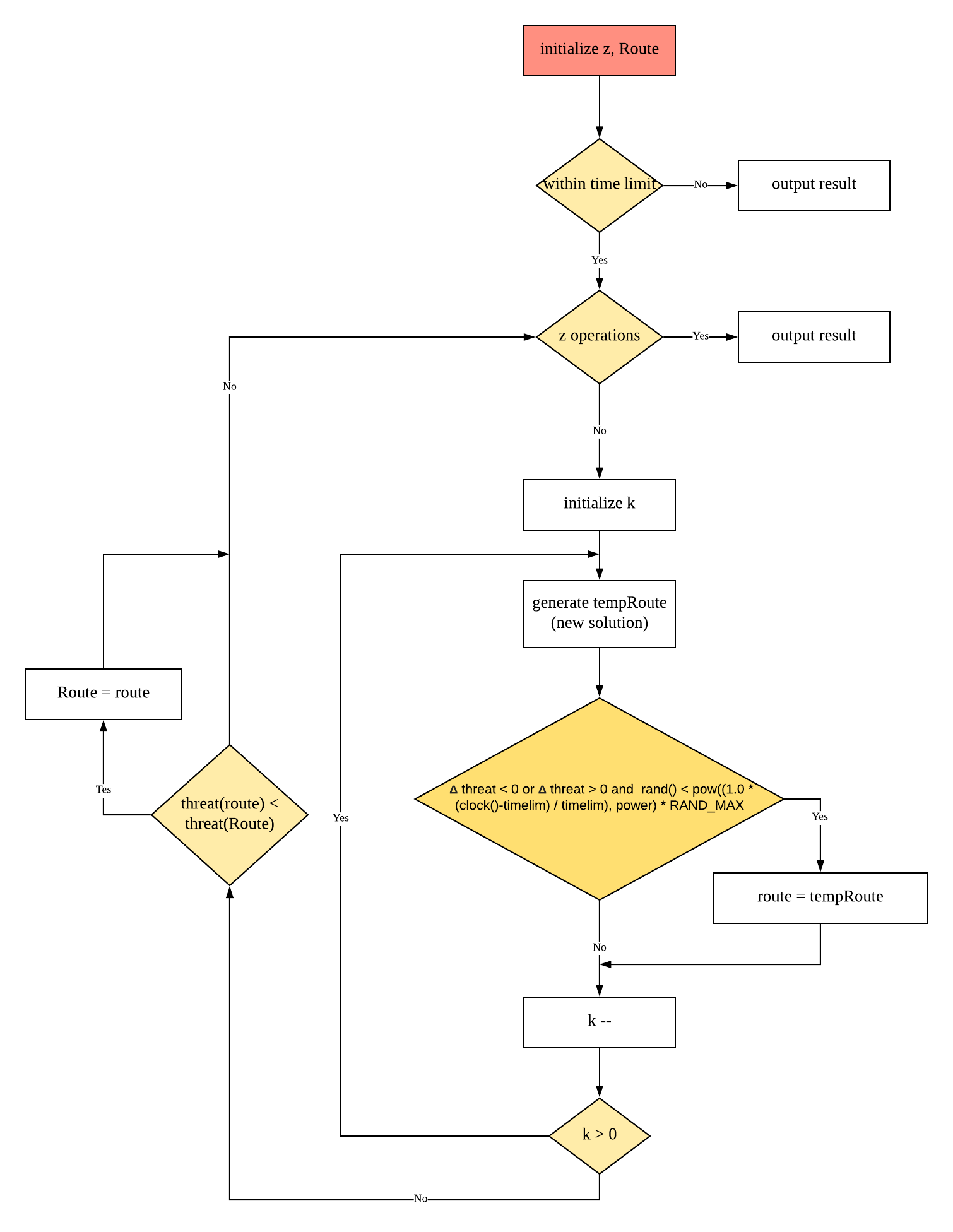
此圖形為簡化過後之判斷函數，x截距為time limit，y截距為RAND\_MAX，可見函數值在x(clock())增加時急遽遞減，符合設計時的要求

1. 當新解被接受，以新解代替當前解進行下次試驗，否則以原當前解繼續進行試驗
2. 若第z輪所得解優於z-1輪，採用此解

**停止準則 :**

最多執行z輪，一輪k次(找新解)，若已達time limit便不再執行下一輪

**流程圖 :**



分工方式

黃心 : 主程式撰寫

黃煒勛 : Github repository 建置&管理、程式排版&註解、書面報告

黃靖翔 : 威脅函數撰寫

呂皓恩 :

心得感想

黃心 :

黃煒勛 :

這次的project雖然是基於前幾次的作業延伸的，但n的增大以及問題的複雜化（加入轉折、時間限制等）使我們無法直接拿之前的code來用，就算要用修改也太過麻煩，不如直接打新的。感謝黃心及某位匿名資工系同學，在project前期就決定好演算法。有一次我們開始亂試兩項參數，結果居然拿了75！？最後發現只會輸出起點附近的隨機點（寫爛了），我們（很有良心）的舉報了，當然淪落到大暴死的下場，之後就在不斷的debug和TLE中度過。另一次則是全部輸出0拿74分，十分神奇。最後截止時雖然不像之前莫名其妙的高分，演算法跟細節也有了很大的改動，不過還行（？總而言之這真的是一個一波三折的project阿！

黃靖翔 :

呂皓恩 :