進階電腦網路

Distance Vector Routing Algorithm

With JAVA UDP Socket

此報告紀錄 S0354008 使用 JAVA(Eclipse)，完成以 UDP 傳送「距離向量路由資訊」之各種事項。

學號(配分)：S0354008(100%)

作者姓名：葉峻嘉

零、目錄

壹、程式功能說明

此程式含以下三項主要功能：

1.Network Initialization：

(1)可手動設定本程式所代表 node 之 port-number：

此功能可加速程式測試，不必每次都查看自動設定而產生的變動 port 值。

(2)可手動設定初始化網路所使用的 txt 文字檔：

此功能可依照使用者輸入的檔名不同而讀取不同網路資料，並且判斷輸入檔案是否存在，不存在則要求重新輸入。

(3)可手動設定本程式所代表之 node：

此功能可依照使用者輸入的 node 名稱不同而扮演不同角色，並且判斷輸入 node 是否於所有 node 中，不存在則要求重新輸入。

(4)可手動設定其餘 node 之 IP、port：

此功能可判斷本程式 node 及其餘 node 的不同，只要求輸入其餘 node 之 IP、port。

(5)設定完成即可列印出本程式 node 與其餘所有 node 的 link cost 表格：

此功能可判斷若 link 不存在，則 cost 為-1。

(6)可透過 socket 傳收 distance vector：

此功能可供本 node 傳 distance vector 給其餘所有 node，並於收到其餘所有 node 的 distance vector 時判斷是否需更新資訊，若需更新，則印出 link cost 表格，準備開始 distance vector routing algorithm。

(7)可使用 distance vector routing algorithm 計算本 node 到其餘 node 之最小 cost：

此功能可在任何 cost 發生改變時，計算新的 least cost 並判斷是否需更新 least cost 及 next hop(同 cost 以字母順序判斷)，若需更新，則印出 link cost 表格，並傳送新的 distance vector 給其餘所有 node；同樣，收到其他 node 送來的，則重複功能(6)，若有需要再重複(7)。

(8)可印出 routing table：

此功能可在 distance vector 穩定後印出此 node 之 routing table(含 destination / next hop / least cost)。

(9)可回到選單：

此功能可在 distance vector 穩定並印出 routing table 約 30 秒後，回到主選單。

2.Link Cost Update：

(1)可手動輸入含有更新 link cost 資訊的 txt 文字檔：

此功能可依照使用者輸入的檔名不同而讀取不同網路資料，並且判斷輸入檔案是否存在，不存在則要求重新輸入。

(2)可檢查哪些 link cost 發生改變：

此功能可印出有更新之 cost 資訊(更新前 / 更新後)並通知其餘 node 前述資訊，也同樣能在收到其餘 node 此類(cost 更新)資訊後，回覆有收到的確認訊息(acknowledge)，更能在收到其餘 node 的確認訊息後印出收到何 node 的確認訊息(acknowledge)。

(3)可在 cost 更新後再次使用 distance vector routing algorithm：

此功能可計算新的 least cost 及 next hop 並印出新的 link cost 表格，且同樣傳送給其餘 node，簡單說，與 Network Initialization 的(6)、(7)功能相同，會不停重複直到穩定。

(4)功能(3)之後的功能，與 Network Initialization 的(8)、(9)功能相同：

穩定後會印出此 node 之 routing table(含 destination / next hop / least cost)，並約 30 秒後，回到主選單。

3.Exit：

(1)30 秒後結束程式，並同時通知其他 node：

此功能可在結束程式之前，傳送訊息給其他 node，表示自己將在 30 秒後結束程式；同樣，收到其他程式的結束訊息會印出來。

貳、程式操作流程(EXAMPLE)

時間

我以老師給的範例說明操作流程(各點一樣的只放一張圖代表)；預設 4 個 node，時間為縱軸：

Node u	Node v	Node w	Node x

Problems @ Javadoc Declaration Co

S0354008 [Java Application] C:\Program Files\Java\

請輸入本機port號以便建立連線：

設 u 為 5555

```
請輸入本機port號以便建立連線：5555
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：
```

設 v 為 6666

```
請輸入本機port號以便建立連線：6666
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：
```

設 w 為 7777

```
請輸入本機port號以便建立連線：7777
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：
```

設 x 為 8888

```
請輸入本機port號以便建立連線：8888
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：
```

皆輸入功能選項"1"，檔名"net1.txt"

```
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：1
請輸入檔案名稱(*.txt)：net1.txt
請輸入此端扮演之node：
```

設為 u

```
請輸入此端扮演之node：u
請輸入 node <v> 之IP：10.91.0.11
請輸入 node <v> 之Port：6666
請輸入 node <w> 之IP：10.91.0.11
請輸入 node <w> 之Port：7777
請輸入 node <x> 之IP：10.91.0.11
請輸入 node <x> 之Port：8888
```

設為 v

```
請輸入此端扮演之node：v
請輸入 node <u> 之IP：10.91.0.11
請輸入 node <u> 之Port：5555
請輸入 node <w> 之IP：10.91.0.11
請輸入 node <w> 之Port：7777
請輸入 node <x> 之IP：10.91.0.11
請輸入 node <x> 之Port：8888
```

設為 w

```
請輸入此端扮演之node：w
請輸入 node <u> 之IP：10.91.0.11
請輸入 node <u> 之Port：5555
請輸入 node <v> 之IP：10.91.0.11
請輸入 node <v> 之Port：6666
請輸入 node <x> 之IP：10.91.0.11
請輸入 node <x> 之Port：8888
```

設為 x

```
請輸入此端扮演之node：x
請輸入 node <u> 之IP：10.91.0.11
請輸入 node <u> 之Port：5555
請輸入 node <v> 之IP：10.91.0.11
請輸入 node <v> 之Port：6666
請輸入 node <w> 之IP：10.91.0.11
請輸入 node <w> 之Port：7777
```

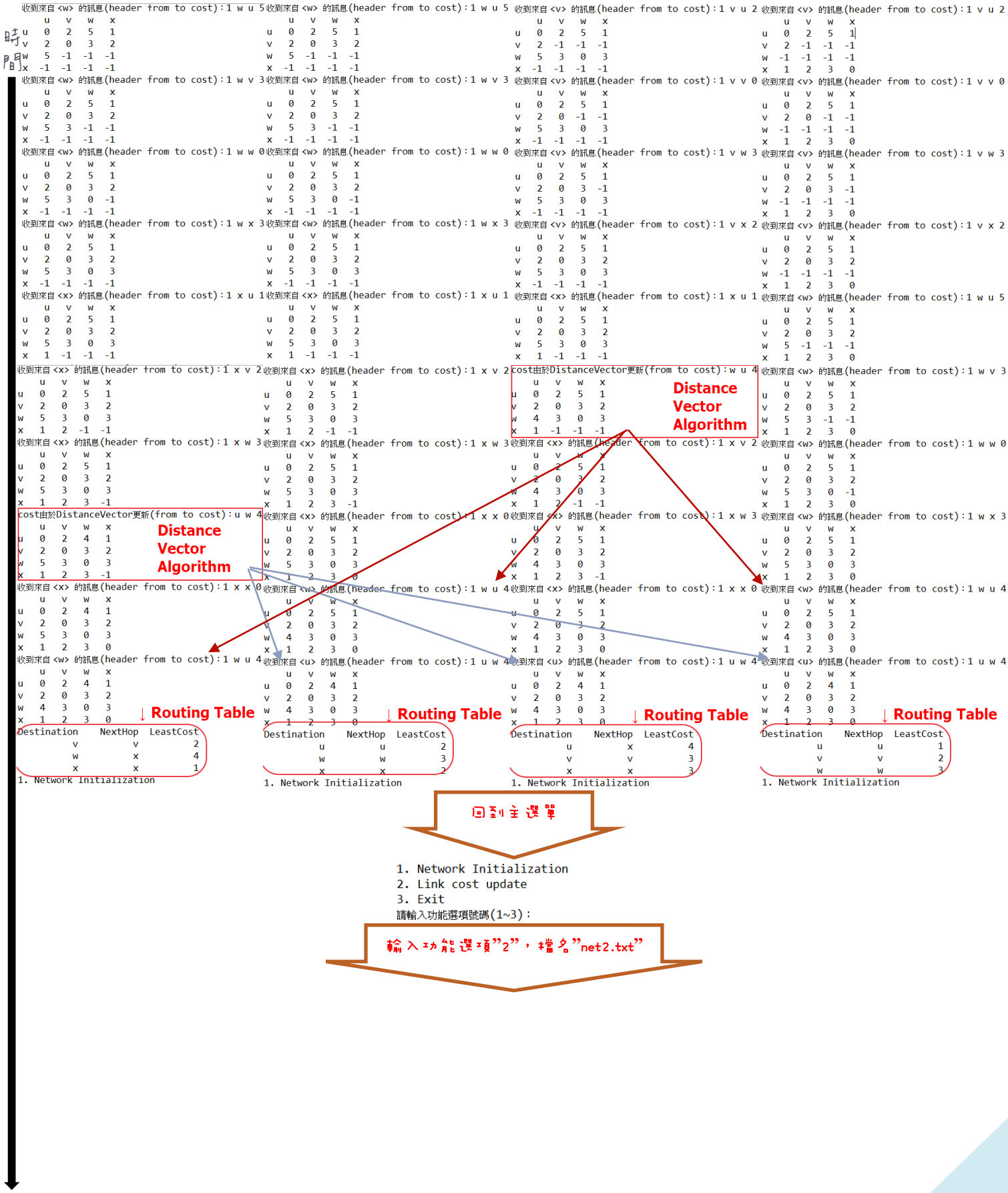
Enter

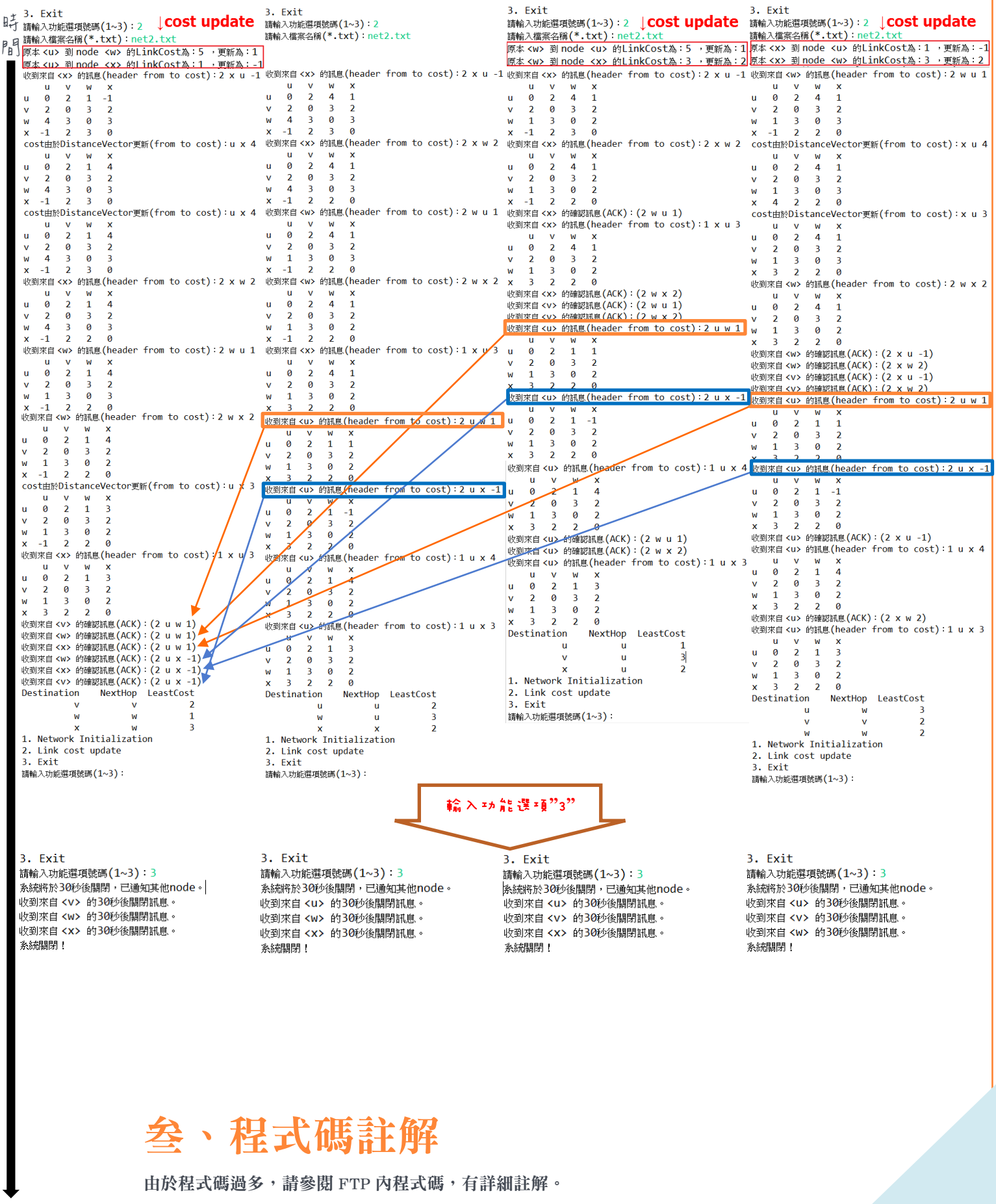
```
請輸入 node <x> 之Port：8888
     u   v   w   x
u    0   2   5   1
v   -1  -1  -1  -1
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <v> 的訊息(header from to cost)：1 v u 2
     u   v   w   x
u    0   2   5   1
v    2  -1  -1  -1
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <v> 的訊息(header from to cost)：1 v v 0
     u   v   w   x
u    0   2   5   1
v    2   0  -1  -1
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <v> 的訊息(header from to cost)：1 v w 3
     u   v   w   x
u    0   2   5   1
v    2   0   3  -1
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <v> 的訊息(header from to cost)：1 v x 2
     u   v   w   x
u    0   2   5   1
v    2   0   3   2
w   -1  -1  -1  -1
x   -1  -1  -1  -1
```

```
請輸入 node <x> 之Port：8888
     u   v   w   x
u   -1  -1  -1  -1
v    2   0   3   2
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u u 0
     u   v   w   x
u    0  -1  -1  -1
v    2   0   3   2
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u v 2
     u   v   w   x
u    0   2  -1  -1
v    2   0   3   2
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u w 5
     u   v   w   x
u    0   2   5  -1
v    2   0   3   2
w   -1  -1  -1  -1
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u x 1
     u   v   w   x
u    0   2   5   1
v    2   0   3   2
w   -1  -1  -1  -1
x   -1  -1  -1  -1
```

```
請輸入 node <x> 之Port：8888
     u   v   w   x
u   -1  -1  -1  -1
v   -1  -1  -1  -1
w    5   3   0   3
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u u 0
     u   v   w   x
u    0  -1  -1  -1
v   -1  -1  -1  -1
w    5   3   0   3
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u v 2
     u   v   w   x
u    0   2  -1  -1
v   -1  -1  -1  -1
w    5   3   0   3
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u w 5
     u   v   w   x
u    0   2   5  -1
v   -1  -1  -1  -1
w    5   3   0   3
x   -1  -1  -1  -1
收到來自 <u> 的訊息(header from to cost)：1 u x 1
     u   v   w   x
u    0   2   5   1
v   -1  -1  -1  -1
w    5   3   0   3
x   -1  -1  -1  -1
```

```
請輸入 node <w> 之Port：7777
     u   v   w   x
u   -1  -1  -1  -1
v   -1  -1  -1  -1
w   -1  -1  -1  -1
x    1   2   3   0
收到來自 <u> 的訊息(header from to cost)：1 u u 0
     u   v   w   x
u    0  -1  -1  -1
v   -1  -1  -1  -1
w   -1  -1  -1  -1
x    1   2   3   0
收到來自 <u> 的訊息(header from to cost)：1 u v 2
     u   v   w   x
u    0   2  -1  -1
v   -1  -1  -1  -1
w   -1  -1  -1  -1
x    1   2   3   0
收到來自 <u> 的訊息(header from to cost)：1 u w 5
     u   v   w   x
u    0   2   5  -1
v   -1  -1  -1  -1
w   -1  -1  -1  -1
x    1   2   3   0
收到來自 <u> 的訊息(header from to cost)：1 u x 1
     u   v   w   x
u    0   2   5   1
v   -1  -1  -1  -1
w   -1  -1  -1  -1
x    1   2   3   0
```


時間
收到來自 <w> 的訊息(header from to cost):1 w u 5
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 -1 -1 -1
x -1 -1 -1 -1
收到來自 <w> 的訊息(header from to cost):1 w v 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 -1 -1
x -1 -1 -1 -1
收到來自 <w> 的訊息(header from to cost):1 w w 0
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 -1
x -1 -1 -1 -1
收到來自 <w> 的訊息(header from to cost):1 w x 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x -1 -1 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x u 1
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 -1 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x v 2
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 2 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x w 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 2 3 -1
cost由於DistanceVector更新(from to cost):u w 4
u v w x
u 0 2 4 1
v 2 0 3 2
w 5 3 0 3
x 1 2 3 -1
收到來自 <x> 的訊息(header from to cost):1 x x 0
u v w x
u 0 2 4 1
v 2 0 3 2
w 5 3 0 3
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w u 4
u v w x
u 0 2 4 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
Destination NextHop LeastCost
v v 2
w x 4
x x 1
1. Network Initialization
收到來自 <w> 的訊息(header from to cost):1 w u 5
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 -1 -1 -1
x -1 -1 -1 -1
收到來自 <w> 的訊息(header from to cost):1 w v 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 -1 -1
x -1 -1 -1 -1
收到來自 <w> 的訊息(header from to cost):1 w w 0
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x -1 -1 -1 -1
收到來自 <w> 的訊息(header from to cost):1 w x 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x -1 -1 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x u 1
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 -1 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x v 2
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 2 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x w 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 2 3 -1
收到來自 <x> 的訊息(header from to cost):1 x x 0
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w u 4
u v w x
u 0 2 4 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
收到來自 <u> 的訊息(header from to cost):1 u w 4
u v w x
u 0 2 4 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
Destination NextHop LeastCost
u u 2
w w 3
x x 2
1. Network Initialization
收到來自 <v> 的訊息(header from to cost):1 v u 2
u v w x
u 0 2 5 1
v 2 -1 -1 -1
w 5 3 0 3
x -1 -1 -1 -1
收到來自 <v> 的訊息(header from to cost):1 v v 0
u v w x
u 0 2 5 1
v 2 0 -1 -1
w 5 3 0 3
x -1 -1 -1 -1
收到來自 <v> 的訊息(header from to cost):1 v w 3
u v w x
u 0 2 5 1
v 2 0 3 -1
w 5 3 0 3
x -1 -1 -1 -1
收到來自 <v> 的訊息(header from to cost):1 v x 2
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x -1 -1 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x u 1
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 -1 -1 -1
cost由於DistanceVector更新(from to cost):w u 4
u v w x
u 0 2 5 1
v 2 0 3 2
w 4 3 0 3
x 1 -1 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x v 2
u v w x
u 0 2 5 1
v 2 0 3 2
w 4 3 0 3
x 1 2 -1 -1
收到來自 <x> 的訊息(header from to cost):1 x w 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 -1
收到來自 <x> 的訊息(header from to cost):1 x x 0
u v w x
u 0 2 5 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w u 4
u v w x
u 0 2 5 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
收到來自 <u> 的訊息(header from to cost):1 u w 4
u v w x
u 0 2 4 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
Destination NextHop LeastCost
u x 4
v v 3
x x 3
1. Network Initialization
收到來自 <v> 的訊息(header from to cost):1 v u 2
u v w x
u 0 2 5 1
v 2 -1 -1 -1
w -1 -1 -1 -1
x 1 2 3 0
收到來自 <v> 的訊息(header from to cost):1 v v 0
u v w x
u 0 2 5 1
v 2 0 -1 -1
w -1 -1 -1 -1
x 1 2 3 0
收到來自 <v> 的訊息(header from to cost):1 v w 3
u v w x
u 0 2 5 1
v 2 0 3 -1
w -1 -1 -1 -1
x 1 2 3 0
收到來自 <v> 的訊息(header from to cost):1 v x 2
u v w x
u 0 2 5 1
v 2 0 3 2
w -1 -1 -1 -1
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w u 5
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 -1 -1 -1
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w v 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 -1 -1
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w w 0
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 -1
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w x 3
u v w x
u 0 2 5 1
v 2 0 3 2
w 5 3 0 3
x 1 2 3 0
收到來自 <x> 的訊息(header from to cost):1 x x 0
u v w x
u 0 2 5 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
收到來自 <w> 的訊息(header from to cost):1 w u 4
u v w x
u 0 2 5 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
收到來自 <u> 的訊息(header from to cost):1 u w 4
u v w x
u 0 2 4 1
v 2 0 3 2
w 4 3 0 3
x 1 2 3 0
Destination NextHop LeastCost
u u 1
v v 2
w w 3
1. Network Initialization
回到主選單
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3):
輸入功能選項"2"，檔名"net2.txt"
Distance Vector Algorithm
Distance Vector Algorithm
Routing Table
Routing Table
Routing Table
Routing Table



叁、程式碼註解

由於程式碼過多，請參閱 FTP 內程式碼，有詳細註解。

肆、心得

1.實作過程心得：

當一開始在課堂上看到這個題目時，還天然地認為難度應該不大。誰知寫的途中，竟然**愈寫愈沮喪**，才發現自己想得太簡單了，好幾次遇到挫折曾想放棄。但好在有著**不放棄的精神**，**化悲憤為力量**，才能克服難關。

從開始研究到完成這個報告與程式碼，**所費約兩周**，其中**每天大概花 6 小時**。每天都寫到**眼睛朦朧混濁**，心態也愈來愈糟糕。就在快要承受不住，想要放棄時(那時半夜兩點)，洗了個熱水澡，告訴自己沒寫出來也問題不大(平時聽從大司馬老師的教誨)，**能寫多少算多少**。從那時才放寬了心，一步一步地冷靜修改，我想此刻必是轉捩點。

程式語言上沒接觸過的東西，你會不確定到底能怎麼用，如此用會不會出錯，**只能不停嘗試**。這其中的過程是最折磨人心的，尤其是這種**牽一髮而動全身的程式**，寫到中期會無從下手。要重頭寫也不是，因為時間不夠；要修改也難辦，不知從何處下手。此種**進退兩難**之局面，真的使我內心百般煎熬。

但經過此事也學到了很多事，例如**心態要保持良好**，屢試屢敗會心情煩躁是在所難免，畢竟我們不是聖人。但要學會休息，**休息是為了走更長遠的路**，這話果然沒錯。在精神已經不佳，心態爆炸的時候，放下工作，睡個覺，洗個熱水澡，或許反而能走得更遠。但這可能也是完成了才敢這樣說吧？

2.問題與討論：

問題(1)：對我來說，此項程式最大的問題，應該就是 **Server、Client 是否要用 while 讓他們無時無刻執行？**因為我在寫完第一部分，要跳到第二部分更新 cost 的時候，發現需要停止 server 功能及 client 功能，才能繼續。但**如何結束**？

討論(1)：我是使用 **socket timeout 的方式設定 Server 的存活時間**，若沒接到新資訊，則會在指定時間後結束，若有新資訊，則重新計算存活時間。但 Client 有點問題，我曾試過讓 Server 結束時將代表結束的 boolean 變數設為 true，以此通知 Client 也要結束，但 Client 結束不了。我推測是 thread 的鍋，Server 一結束，計算資源全部輪到 Client 手中，根本無法執行將 boolean 變數改變的指令，自然 Client 也結束不了。

我的解決方式是**不要一直用 while 執行 Client**，改成**要送再送**，但這其中也是一波三折，嘗試好幾次才成功，大致上是**每次要送的時候都 new 一個新的 Client 物件**。

問題(2)：Exit 的時候，只輸入一個選項 3 馬上跳到 Exit 功能開始傳送結束資訊，我**無法讓後輸入選項 3 的 node 接收到先輸入選項 3 的 node 的程式關閉資訊**。

討論(2)：因為我是需入選項 3 以後，才開始新的 Server 啟動，新建 Socket。解決辦法其實很簡單，就是將 **Server 要用的 Socket 宣告成 public class S0354008 底下的 static 變數**，並且在結束程式之前都不要 close 該 Socket，**只在最後一個功能 close 就好**，如此一來 **Socket 的 Buffer 都一直存在**，自然也就能接到東西。

問題(3)：第二功能的 **ACK 訊息**一開始我會 LOSS 掉。

討論(3)：這是因為**執行緒配合的問題**，有可能你收到一個 cost 更動的資訊，要回傳 ACK 時，**還沒回傳，馬上又收到另外一個，由同一個點送來的更動資訊**，這時你會以為只要 ACK 一次。解決辦法是當收到更動資訊時，啟動 Client 先送把第一筆 ACK 資訊送掉，然後 **Server.sleep()**，這樣 Server 就不會連續收到 2 個更動資訊而以為只有一個了。

伍、延伸功能

1.Node 陣列為「動態陣列」，換言之，不論 node 個數皆適用：

```
public class S0354008 {
    static ArrayList<NetNode> allnodes = new ArrayList<NetNode>(); // 動態陣列，儲存所有node，可變動長度，不固定。
    static char HostNode; // 本機扮演的node。
    static int HostNodeNum; // 本機扮演的node在動態陣列內的index。
    static int HostPort; // 本機port號。
```

2.判斷功能選項號碼是否符合，並提供重新輸入：

```
請輸入本機port號以便建立連線：5555
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：4
輸入錯誤，請重新輸入。
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：
```

```
輸入錯誤，請重新輸入。
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：aaaaa
輸入錯誤，請重新輸入。
1. Network Initialization
2. Link cost update
3. Exit
請輸入功能選項號碼(1~3)：
```

3.判斷輸入檔案是否存在，並提供重新輸入：

```
請輸入功能選項號碼(1~3)：1
請輸入檔案名稱(*.txt)：aaaa.txt
檔案不存在，請重新輸入。
請輸入檔案名稱(*.txt)：
```

4.判斷 node 是否存在，並提供重新輸入：

```
請輸入功能選項號碼(1~3)：1
請輸入檔案名稱(*.txt)：net1.txt
請輸入此端扮演之node：g
無此node，請重新輸入：
```