Problem 5

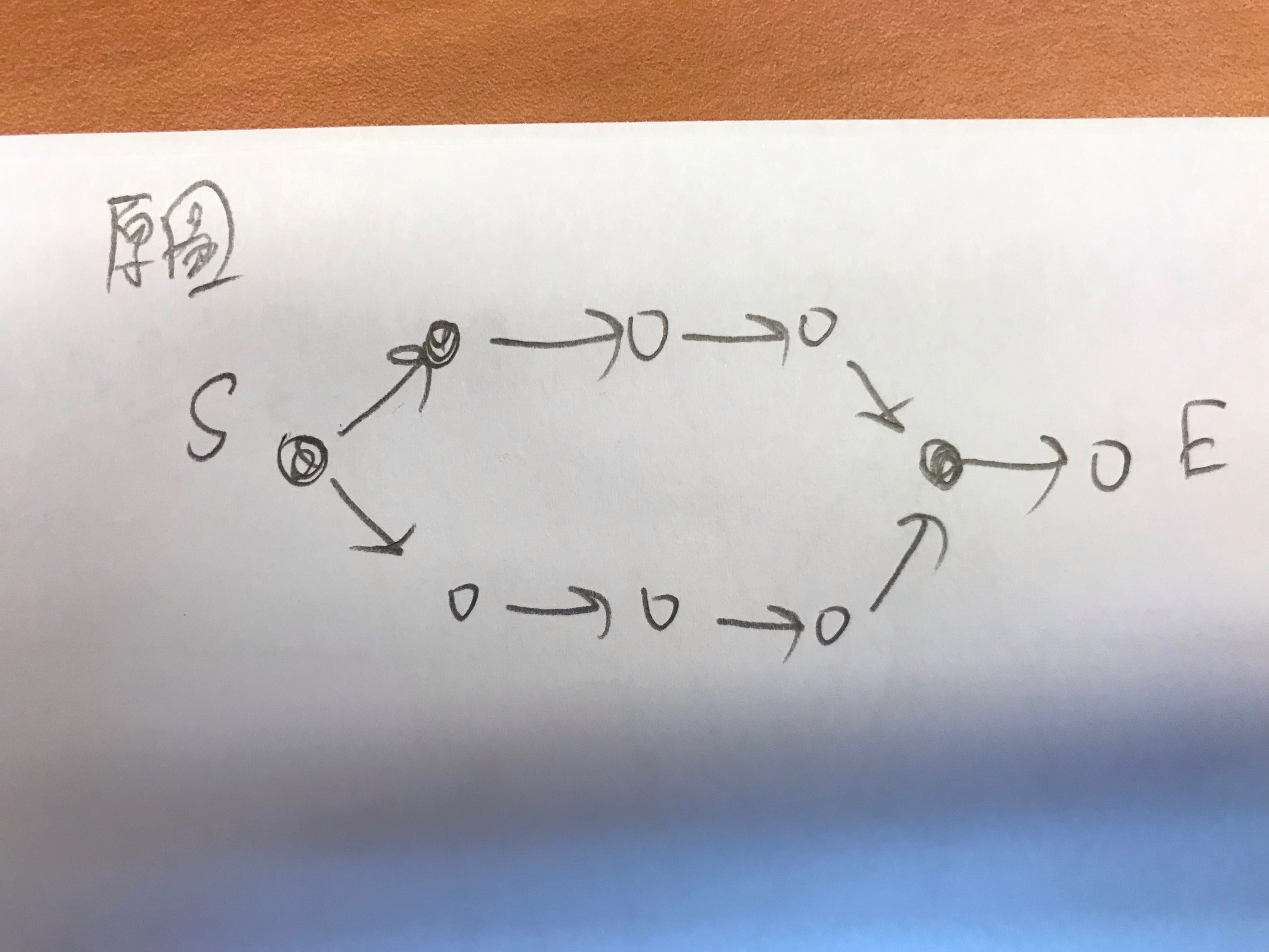
5-(a)

1設原本有生命值HP=3(一開始就吃到pie，只能再多走2個城市)

2對原圖複製兩份verteces，成為一個有3\*|V|的大圖，大圖可以分成3個區塊，HP=3, HP=2, HP=1。

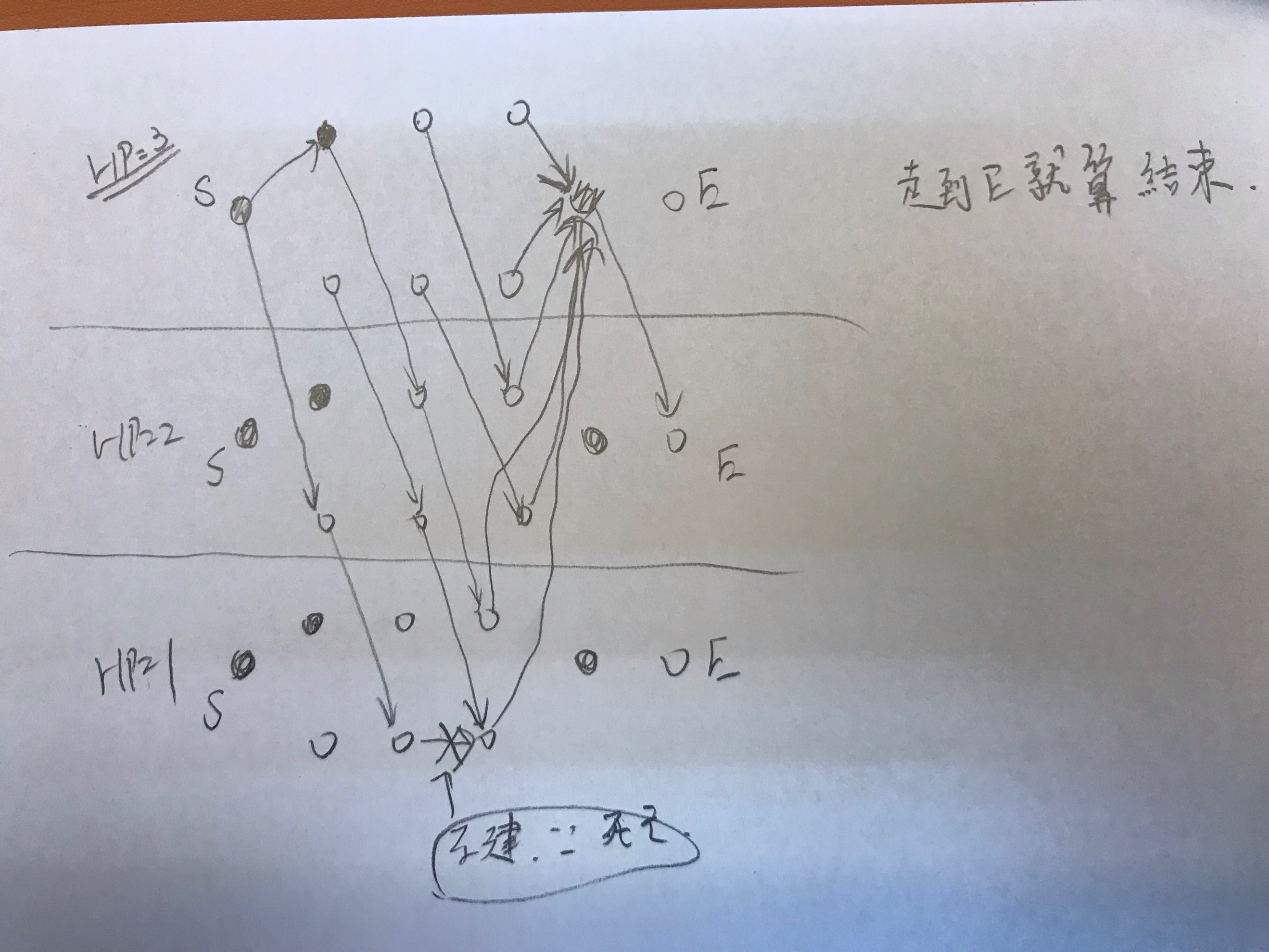
3 開始建edge，建edge的方法是走訪所有的vertices。如果吃到pie，是從HP=3的那部分開始走。如果原圖的下一步是有pie的城市，就建一條邊回到HP=3的部分，如果原圖下一步是沒派的城市，就建一條邊到HP少1的那一部分圖

4針對大圖的empire city做dijkstra



大圖中會有3\*|V|個vertex和O(3\*|E|)條邊。

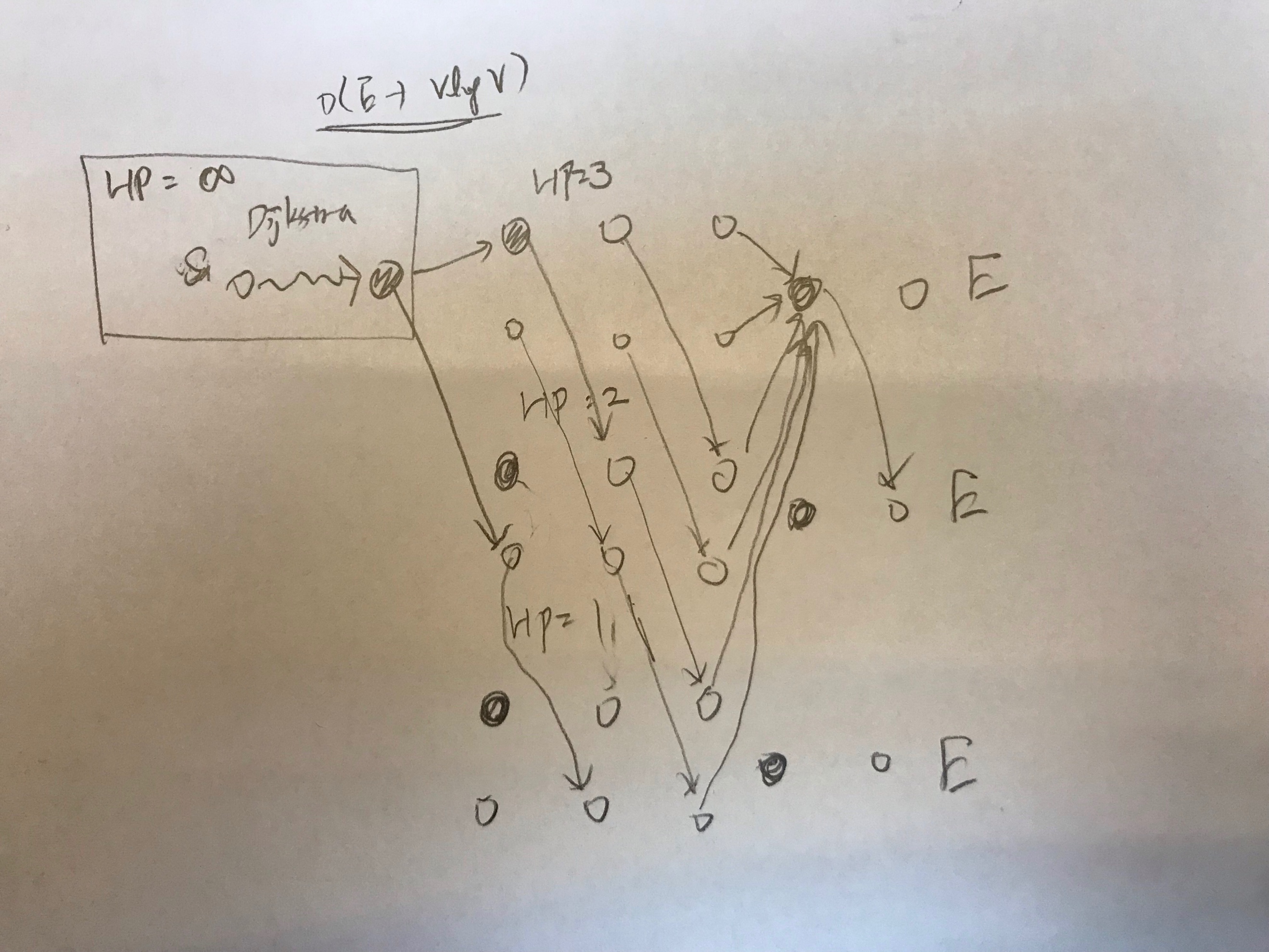
第3步花3\*O(V+E)，第4步花O(Dijkstra(V’=3\*V,E’= 3\*E))，一共花3\*O(V+E)+O(3\*Dijkstra(V,E))=O(Dijkstra(V,E))。



5-(b)

假設一開始沒吃到pie，一開始就對原圖做dijkstra，若一路上都沒遇到有pie的城市，直接return。(直接得到正解)

若走到有pie的城市，重複5-(a)，複雜度仍為O(Dijkstra(V,E))



5-(c)

1 設原本有生命值HP=k+1

2 針對原圖的empire city做dijkstra，若從頭到尾都沒吃到pie就直接return正解。

3 若一開始還沒吃到pie，但中間吃到pie，就走到吃到pie為止，對吃到pie之後的原圖複製k份verteces，成為一個有O(k\*|V|)的大圖，大圖可以分成k個區塊，HP=k+1, HP=k,…, HP=1。

4 開始建edge，建edge的方法是走訪所有的vertices。如果吃到pie，是從HP=k+1的那部分開始走。如果原圖的下一步是有pie的城市，就建一條邊回到HP=k+1的部分，如果原圖下一步是沒派的城市，就建一條邊到HP少1的那一部分圖

5 對大圖的HP=k+1部分的原點做dijkstra，得到解答。

大圖中會有(k+1)\*|V|個vertex，O((k+1)\*|E|)條邊。

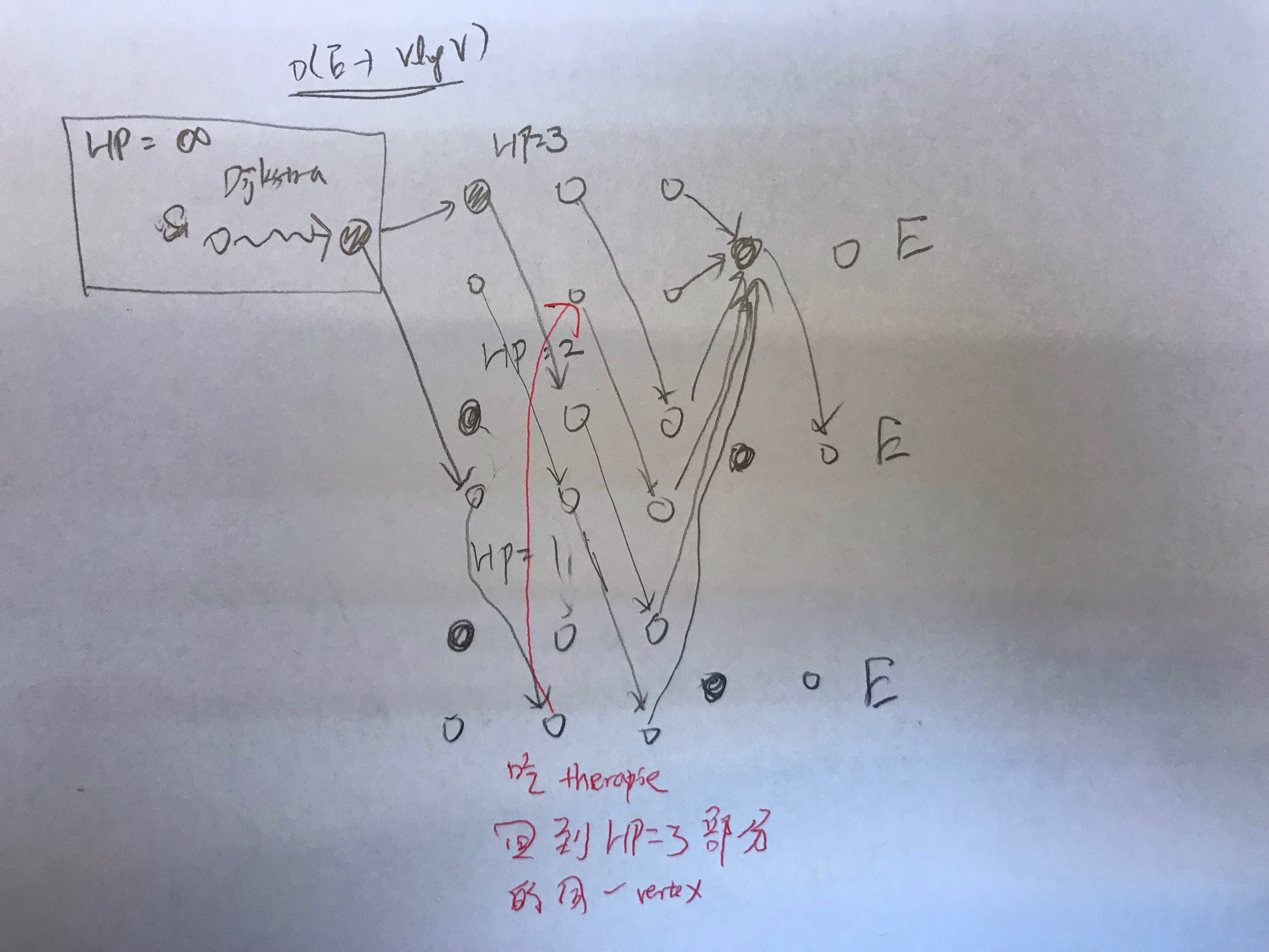
第3,4步建圖花(k+1)\*O(V+E)，第5步花O(dijksra(V’=(k+1)\*V, E’=O(k+1)\*E) =O(k\*dijkstra(V,E))，總共花O(k\*dijkstra(V,E))。

5-(d)

設theraPIE=True

前面作法和5-(c)一樣，但是在HP=1部分，吃theraPIE(theraPIE=False)使得HP補滿，回到HP=k+1部分的同一個vertex繼續走。若可以走到終點則發現shortest-path，但若還是走不到終點，只好回到吃theraPIE的當時的前一個vertex，同時theraPIE=True，重找一條路做dijkstra，不斷重複到找到終點為止。

以上只比5-(c)增加了一個變數theraPIE，吃了theraPIE可能讓本來不能走到的路變得走得到，也可能讓本來沒辦法走到的路花更多的時間才發現不能走。但是整體而言，對於演算法的時間複雜度影響為O(1)，仍為O(k\*Dijkstra(V,E))。



5-(e)

5-(f)

選擇的策略是：一遇到有pie的城市就淨化。

總共可以淨化k2個城市。

除了複製k份原本的vertex外，在另外複製k2份原本的vertex，做一張超級大的graph。多複製k2的那部分，代表所剩下的淨化額度。

也就是現在除了HP=k+1之外還有一個是MP=k2。所以大圖上會有HP=k, k-1, …,1的部分還有MP=k2, k2-1, …,1的部分。

一開始從MP=k2的部分開始做dijkstra，走到有pie的城市發動淨化，就會到MP=k2-1的部分，直到MP=0就會進入HP=k+1的部分。之後就會像5-(c)。

現在的graph V’=(k+1)\*V+ k2\*V E’= O((k+1)\*E+k2\*E)，所以總共需要O(Dijkstra(V’, E’))=O((k+k2)\*Dijkstra(V,E))

5-(g)

每一個時間點都有可能可以發動Nirvana，所以要針對每一個vertex，另外再多k2個vertex。

同樣使用5-(c)的建圖方法，只是現在針對大圖上面每一個vertex，多畫k2個vertex，這k2個vertex是原圖的subgraph。建立一個更大的graph。

如此會有(k+1)\*|V|\*(k2)的vertex，邊會有O((k+1)|\*|E|\*k2)條。

對超大圖的原點做dijkstra，所花的時間複雜度為O(dijksra(V’=(k+1)\*V\*k2, E’=O((k+1)\*E\*k2)))=O((k\*k2\*dijkstra(V, E)))。

以下的圖**只有針對一個vertex做Nirvana**，沒有對每一個vertex做Nirvana。

