ADA HW3

Problem 5

//和李長諺、陳柏諺、沈立程、石容居討論

(1)

$deg(r)$就是答案
因為T是G的DFS tree而r又是T的root
由Theorem 22.10可知DFS tree中只有tree edge和back edge
因為沒有cross edge所以和r底下的子樹必經過r才能走到其他子樹
因此子樹們必會在不同connected component，而子樹數量為deg(r)

//和李長諺、陳柏諺、沈立程、石容居討論

(2)

$deg(v)$就是答案
由Theorem 22.10可知DFS tree中只有tree edge和back edge
又因為v的祖先和子孫間無邊，因此T中沒有back edge
令k的child代表點k不含ancestor的鄰居，k的子樹代表包含k的某一個child的descendant，k的parent代表點k的ancestor且和k有邊
令t表示T中除了v的descendant和不為root且為v的ancestor以外的所有點

因為DFS tree必不包含cross edge，所以v的任意兩不同子樹間必不相連
而v的子樹和v的ancestor也必不相連，否則形成back edge
而t和v的任意一個子樹也不相連，若相連也會形成cross edge

因此v的子樹和t以及子樹間一定不存在相連的邊，因此
s(v,G)=deg(v)

//和李長諺、陳柏諺、沈立程、石容居討論

(3)

$k+1-\Sigma_{i=1}^{k}(up_T(w_i)<depth(v))$
已知若$D_T(w_i)$不和v自己外的ancestor相連的話，則
$depth(v) \leq depth(u)\ for\ all\ u \in D_T(w_i)$

如果$up_T(w_i)=depth(v)$代表$D_T(w_i)$中沒有邊連接到不含v的v的ancestor，所有的邊都在$D_T(w_i)$和v中互相連接，而
$depth(u)=depth(v), u \in D_T(v)$發生在u為v的child時
如果$up_T(w_i)<depth(v)$代表$D_T(w_i)$中出現連接到v自己外的ancestor的edge
因此$D_T(w_i)$就會和v自己外的ancestor算在同一個connected component中，因此要扣掉
當$N_G(v)$都沒有相接的邊時$s(v,G)=deg(v)=k+1$
因此所求為$k+1-\Sigma_{i=1}^{k}(up_T(w_i)<depth(v))$

//和李長諺、陳柏諺、沈立程、石容居討論

(4)

首先選定一個點r作為root，接著開始DFS紀錄所有點的最小深度，至此為$O(|V|)$
接著再遍歷所有點紀錄每個點的descendants的最小深度，對點v而言
$up(v)=min(up(w_1),up(w_2)...up(w_k),depth(v))$
要走過所有點和邊所以是$O(|V|+|E|)$
接著對於不是root的點利用(3)的方式看每個點v的child w_i擁有的$up(w_i)$並算出$s(v,G)$
而root的話則是$deg(root)$

因為這也要走過所有邊和點所以時間複雜度為$O(|V|+|E|)$
總共為$O(|V|+|E|)$

Problem 6

//和李長諺、陳柏諺、沈立程、石容居討論

(1)

Kruskal’s algorithm相同，只是把min heap換成max heap
排序所有邊是$O(E\ logE)$，Disjoint set為$O(E\alpha(V)) \approx O(E)$
時間複雜度是$O(E+E\ logE) = O(E\ logE) = O(E\ logV)$
因maximum spanning tree也有cut property，只是邊e換成連接兩集合的最大邊
而透過改良後的Kruskal algo每次取的點的也是連接已取、未取點的最大邊
且取出的邊的數量也和maximum spanning tree相同，因此這個方法可以找到maximum spanning tree

//和李長諺、陳柏諺、沈立程、石容居討論

(2)

假設圖中有環，則環中最窄的邊不會存在於maximum spanning tree中，因為假設環中最窄邊存在於maximum spanning tree中，則移除他後會使得tree變成兩個component，而這兩個component之間必有另一條更寬的邊存在，因此矛盾

假設對於點對s、t，maximum spanning tree中不包含他們之間widest的path
也就是maximum spanning中包含了某條邊比widest path中最短邊還窄
那麼當把這條widest path加進maximum spanning tree時，tree中會形成環

而環中最窄的邊不會存在於maximum spanning tree中，但此時環中最短的邊是maximum spanning tree中s到t最短的邊，因此產生矛盾所以maximum spanning tree必包含某條widest path

//和李長諺、陳柏諺、沈立程、石容居討論

(3)

假設當s到v和u的最短路徑都不終於(u,v)，以 $dis(a,b)$和$dis^{'}(a,b)$表示在修改前後從a走到b的最短距離，$len(P)$和$len^{'}(P)$表示P在修改前後的長度，$<a,b\dots c>$表示路徑

令S_1代表從s出發經過(u,v)走到v的最短路徑
令S_2代表從s出發經過(v,u)走到u的最短路徑
可以得到$len(S_1)\geq dis(s,v)+\epsilon_1$和$len(S_2)\geq dis(s,u)+\epsilon_2$
取$\epsilon=min(\epsilon_1,\epsilon_2)$

假設L是修改過的圖上一從s走到$w(w\in V)$的最短路徑，會得到L有三種case
case 1：(u,v)在L上
$$dis^{'}(s,w)=len^{'}(L)=len^{'}(<s\dots v>)+len^{'}(<v\dots w>)$$

$$\begin{aligned}
&=len^{'}(<s\dots v>)+len(<v\dots w>)\\
&=len(<s\dots v>)-\epsilon+len(<v\dots w>)\\
&\geq dis(s,v)+\epsilon-\epsilon+dis(v,w)\\
&\geq dis(s,w)
\end{aligned}$$

又因為把路徑改小一定不會使最短路徑長增加，因此
$dis^{'}(s,w)\leq dis(s,w)$
因此$dis^{'}(s,w)=dis(s,w)$

case 2：(v,u)在L上
和case 1同理，可用相同方法得到$dis^{'}(s,w)=dis(s,w)$

case 3；(v,u)、(u,v)都不在L上
因為(v,u)、(u,v)都不在L上，因此減去\epsilon也不會改變最短路徑長因此dis^{'}(s,w) = len^{'}(L)=dis(s,w)$

若，則當$d((u,v))$減少ϵ $(\epsilon > 0)$時

已知邊(u,v)在s到u或v的shortest path上，假設新shortest path不包含(u,v)
已知原本任何其他的path都≥原本的shortest path，因此更新(u,v)後的所有不含(u,v)的path都會≥更新後的原本的shortest path，因此更新後的shortest path必包含(u,v)因此一定會變小
因此(u,v) downward critical

∴ A road (u, v) ∈ E is downwards critical if and only if there is a shortest
path from s to u or v that ends at (u, v).

//和李長諺、陳柏諺、沈立程、石容居討論

(4)

(u,v) is upward critical iff s到v的所有shortest path都會經過(u,v)

若s到v的所有shortest path都經過(u,v)，則當增加(u,v)時因為s到v的所有shortest path都會經過(u,v)，因此原本的shortest path distance一定都會變大
而除了原本shortest path的其他所有path的distance必大於原本的shortest path distance，因此若s到v的所有shortest path都會經過(u,v)，則(u,v)會是upward critical

假設(u,v) upward critical且s到v的shorest path至少一條不經過(u,v)且長度為b
而s經過(u,v)到v的最短路徑長度為a，且$a \geq b$，則對(u,v)增加$\epsilon(\epsilon > 0)$後，$a + \epsilon > b$
而此時仍有一條路徑為原來的shortest path長度為b，因此最短路

徑長不變所以不為upward critical，矛盾，所以s到v的所有shortest path都會經過(u,v)

$\therefore$ (u,v) is upward critical iff s到v的所有shortest path都會經過(u,v)

//和李長諺、陳柏諺、沈立程、石容居討論

(5)

利用 Dijkstra’s algorithm計算shortest path，還要對每個點各開一個vector紀錄符合的邊
利用陣列w紀錄當前走到點w_i的最小值
當從priority queue中取出點v且更新到邊(v, u)時
如果$w_v + d(v, u) < w_u$，則清空$vector_u$然後把v放進去，接著更新$w_u = w_v + d(v, u)$
如果$w_v + d(v, u) = w_u$，則把v放進$vector_u$

這樣就可以記錄到所有在shortest path中的邊
而當(u,v) downwards critical時會有shorest path end在u或v
而最短路徑的subpath都是最短路徑，因此上述vector中的邊都downwards critical

而當(u,v) upward critical時，s到v的所有shortest path都會經過(u,v)
而剛剛的vector利用是哪個點走到自己紀錄shortest path上的邊
所以當點v的vector中只有一個元素時，代表所有走到點v的shortest path都會經過這條邊
因此這條邊會滿足upward critical的條件，只要對每個vector都看一次就行

Prove of correctness：

拆點時會把點v拆成兩點一邊，定義這兩點為A,B兩種點使得同點拆完後的邊(A,B)權重為$-k(v)$，下面用點A表示A這種點，點B表示B這種點，原本的圖為G、拆點後的圖為G'

點B的前一個一定是點A且點A一定指向點B，因此點B的下一個一定指向點A，否則會出現點B由點B指來的情況。此外點A的前一個一定會是點B，否則會出現點A指向點A的情況

因此可以知道在G'中所找到的環一定是ABAB間格出現，因此一定可以把G'中找到的環對應到G中唯一的環。而G中的環透過拆點後也能在G'中找到唯一對應的環，因此可以得到G和G'的環為一對一

接著對原先的點G做操作，把所有的邊權重乘上K並把點權重乘上負號
由題目給的不等式可以得到只要在改過的G上找到負環即是存在所求的環，而在G'中由B指向A的邊權重和原本的邊一樣，由A指向B的邊權重即是原本的點權重，且G中的點拆點後A指向B的邊唯一。因此在G'中找到的負環一定可以對應到G中的某個負環，故這就是所求

Time Complexity Anaylis

利用Fibonacci heap實作Dijkstra’s algorithm找shortest path tree會花費$O(E + VlgV)$
而對額外的vector操作每次insert會是$O(1)$，clear可以$O(1)$實作
因此對vector操作的複雜度最糟是$O(E)$

而尋找downwards critical的邊會是把所有vector中紀錄的邊找出來因此最糟是$O(E)$
尋找upward critical的邊會看每個點的vector大小是否為1，只有一條邊所以標記也是$O(1)$
每次check size是$O(1)$，總共是$O(V + V) = O(V)$
因此總複雜度會是$O(E + V + E + VlgV) = O(E + VlgV)$
又因為圖connected所以$E > V$，因此複雜度會是$O(ElogV)$

//和李長諺、陳柏諺、沈立程、石容居討論

(6)

令存圖和權重的方式為adjacency list，由vector實作
因為邊有向且為SCC，所以先對原始V個點拆點
每次取出任意一點v $v \in V$時都先建立一空點w，接著把所有v上連接的邊(v, n_i)都改成(w, n_i)，也就是改由w接出去，且原本v上的邊也要丟掉，這些邊的權重都要乘上K再由w紀錄起來
最後建立一條邊(v,w)且權重為$-k(v)$，最後再繼續拆下一個點

如此可以把圖變成一張有2V個點，V+E條邊的圖
題目要求是找出$\frac{\Sigma_{i=1}^{n} k(v_i)}{\Sigma_{i=1}^{n} d(v_i, v_{i+1})} > K$，而邊都已經被reweight過一次
因此現在的目標變成找出這張圖是否有負環，因為現在點權重也已經被拆成邊
題目要求的條件變成$\Sigma_{i=1}^{n} d(v_i, v_{i+1}) * K - \Sigma_{i=1}^{n} k(v_i) < 0$

找負環可以利用Bellman-Ford algorithm的方式
改成找到負環return True，否則return false

每次建立空點和重建一條邊及其權重是$O(1)$
拆點時要走過所有點和邊並重建每一條邊，因此時間複雜度是$O(V + E)$
而Bellman-Ford algorithm找負環所需時間為$O(VE)$
因此總共需要$O(VE + V + E) = O(VE)$