

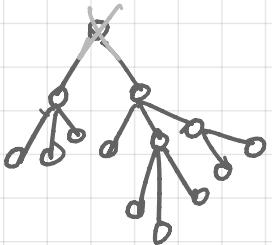
Problem 5

(1) 因為 tree 沒有 cycle, 所以移除掉 root 之後 該是 root 的 children
皆不是 root 的 sub-tree 並不會彼此相連 所以移掉 root 會產生

$\# \text{ of root's children} = \deg(r)$ 個 sub-tree, which are connected

components.

$$S(r, T) = \underline{\deg(r)} \rightarrow *$$



Problem 5

(2) 假設原本的 graph G 是 connected, 拆掉 v 後得到 G' . 原本的 DFS tree T 在 T 中, 定義 type① subtree 由 v 和 $v.parent$ 相連的 vertices 組成
type② subtree 由 v 和某一 $v.child$ 相連的 vertices 組成

- 若 type① 和 type② 在 G 中未相連, 2 不同 child 組成的 type② 在 G' 中未相連

則拆掉 v 會使 G' 有 $\# \text{of type}① + \# \text{of type}② = \deg(v)$ 個 connected components

<Claim> 會不存在 $\{u, w\} \in E$, where $u \in \text{ancestor}(v)$, $w \in \text{descendant}(v)$ 且

(1) type① 和 type② 在 G' 中未相連, type② 之 p_f 在 G' 中相連

- $p_f(1)$ type① type② 未相連

由反證法, let $w \in \text{child}(v)$

if $\exists u \in \text{descendant}(w)$

$\exists y \in \text{descendant}(v)$ (^{and} $y \in \text{ancestor}(v)$)

and $\exists \{u, y\} \in E$, 因 $\text{edge}(u, y)$ 在 DFS-algorithm 中, 只可能是 tree edge or back edge.

\therefore 兩種 case - 1. $y = \text{ancestor}(u) = \text{descendant of } (v)$ 且 $y \neq \text{ancestor}(v)$
 \rightarrow both contradict ($\rightarrow \Leftarrow$)

2. $y = \text{descendant}(u) = \text{descendant }(v) \rightarrow$ contradict ($\rightarrow \Leftarrow$)

- $p_f(2)$, type②, type② 未相連

由反證法, let $w_1, w_2 \in \text{child}(v)$, $w_1 \neq w_2$

if $\exists u_1 \in \text{descendant}(w_1), u_2 \in \text{descendant}(w_2)$

and $\exists \{u_1, u_2\} \in E$, 則在 DFS-tree ϕ , 要麼 $u_1 = \text{descendant}(u_2)$

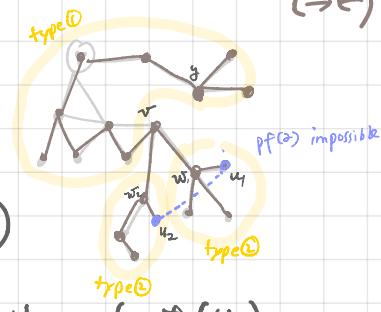
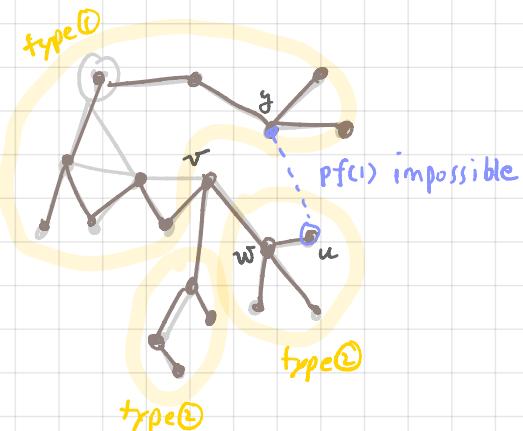
要麼 $u_2 = \text{descendant}(u_1)$

\rightarrow both are impossible. 因為 u_1, u_2

是 2 sibling 旗下子 descendant,

它們至 T 中只有可能已經過 v 相連

共同 ancestor



(3) 連續上題定義的 type①, type②, 由 $PF(2)$ 已知任何條件下
type②之間不可能存在 G 中相連

<Claim> type①和由 w 組成的 type②在 G 中相連 ($w \in \text{child}(v)$)

$$\Leftrightarrow \text{up}_T(w) < \text{depth}(v)$$

Pf " \rightarrow " if type①和由 w 組成的 type②在 G 中相連 ($w \in \text{child}(v)$)

$\exists y \in \text{type}①, u \in \text{type}② = \text{descendent}(w), \exists \text{edge}(y, u)$
in G .

$\therefore y$ must $\in \text{ancestor}(v)$ or there would be cross edge between
 y and u .

$$\therefore \text{depth}_T(y) < \text{depth}_T(v)$$

$$\text{X } \text{up}_T(w) = \min_{y \in N_G(\text{P}_T(w))} \text{depth}_T(y) < \text{depth}_T(v) < \text{depth}_T(w)$$

Pf " \leftarrow " if $\text{up}_T(w) < \text{depth}(v)$,

$\because \min_{y \in \text{P}_T(w)} \{\text{depth}_T(y)\} > \text{depth}(v), \therefore$ 不存在 $u \in \text{descendant}(w)$

和某 $y \in \text{descendant}(w)$
相連. 又 \because 沒 cross edge.

$y \in \text{ancestor}(v) \in \text{type}①$

因為相連的 type①, type②會在 G 中形成 connected component,

不相連的 type①, type② subtree 在 G 中各自是 connected components,

又由<Claim>, 可知 $\{\# \text{of 相連的 type}①, \text{Type}②\} = \{\# \text{of } \{w \mid w \in \text{child}(v) \text{ and } \text{up}_T(w) < \text{depth}(v)\}\}$

$$\therefore G \# \text{of connected components} = \deg(v) - \#\{w \mid w \in \text{child}(v) \text{ and } \text{up}_T(w) < \text{depth}(v)\}$$

(4) 算 DFS 時，每個 node u 額外記錄一個變數 $u.p$ ，並記得 $depth_{\tau}(u)$
in DFS tree

$$\text{執行完要得到: } \begin{cases} u.up = u.p(u) \\ u.d = depth_{\tau}(u) \end{cases}$$

- 在 DFS recursively call next DFS to explore 子孫，傳入 current $depth + 1$ 。參數，每個 node 在 discover 時，將 $u.d$ 設成 i^2 $depth$ ，並 initialize $u.up$
- (1) - 在 traverse 過程中，每次遇到 back edge v , (i.e. meet some node already discovered)
 - 若 $v.d < u.up$, 更新 $u.up = v.d$.

- (2) - 每次 recursively call next DFS to explore child node 孫子，會回傳 child w 的 $w.up$ ，若 $w.up < u.up$, 更新 $u.up = w.up$.

<Claim> 由上述，當以 u 發的 DFS 結束，確保 $u.up = u.p(u)$

<PF> By induction of height (DFS-tree rooted by node u)

- base case: height = 1, node u 的 $D_{\tau}(u) = \{u\}$

在 $DFS(u)$ 的過程中，所有含 u 的 edge 都會檢查 $depth$ 並更新 $u.up$
不難得 $u.up = \min_{y \in N_G(u)} depth_{\tau}(y)$

- inductive case: height = k , root u 的 $D_{\tau}(u) = \{D_{\tau}(w) \mid w = \text{child}(u)\} + \{u\}$

$$\therefore u.p(u) = \min_{y \in N_G(D_{\tau}(w))} depth_{\tau}(y)$$

$$= \min \left\{ \min_{y \in N_G(u)} depth_{\tau}(y), \min_{y \in N_G(D_{\tau}(w)) \setminus \{w\}} depth_{\tau}(y) \right\}$$

在 $DFS(u)$ 過程中，
都會檢查所有相連 node
與 u
by depth (上面第二行)

\downarrow \downarrow
 $= \min \{ u.p(w) \mid w \in \text{child}(u) \}$
 \downarrow
 在 $DFS(u)$ 過程中，recursively
call $DFS(w_i)$ 會回傳 $w.up$

by induction hypothesis
 $w.up = up_T(w)$
 (上面未作(2))

- 上述的 algorithm 是在既有的 DFS-algorithm 上增加幾個 $O(1)$ operations, 沒有增加複雜度. 使得對於所有 node $u \in up_T(u)$ $\sim O(|V|+|E|)$
- 皆下來, 可以用 (3) 小題的方法計算所有 node u 的 $s(u, G)$,

$$s(u, G) = \deg(u) - |\{w \mid w \in \text{child}(u) \text{ and } w.up < \text{depth}(u)\}|$$

$\sim O(1)$ $\sim O(\deg(u))$.

其中, 展開 child, 可以 traverse over adjacent list, 若 $w.parent = u$ 就是.

→ 所有 u 皆找其 $s(u, G)$ 的 cost $\sim O(2^*|E|) = O(|E|)$

totally $O(|E| + |V|)$

Problem #6

(1) 將 Prim's algorithm 中用的 min-queue 替換成 max-queue [將原圖反過來] 並證明 maximum spanning tree

- Modified cut property: let C be the cut in a graph,
 e is edge across C w/ maximum cost,
then maximum spanning tree contains e

\leftarrow 簡化問題 (for simplicity, 這裡指 e unique)

- remove e , 剩 max-ST (maximum spanning tree) 成 2 components T_1, T_2
 - $\exists e'$ reconnect $T_1 \& T_2$ into T'
 - $\because \text{weight}(e') > \text{weight}(e)$, new tree T' has bigger weight than T ($\rightarrow \Leftarrow$)
- 因為 modified Prim's algorithm 每次都會新增和 current tree 相連且一端 $\in T$, 一端 $\notin T$ 的 edge e , weight 最大 by edge e
 - \Rightarrow modified cut property, $e \in \text{max-ST} \rightarrow \text{最後的 } T \subseteq \text{max-ST}$
- Time complexity 和 Prim's algorithm - 不同 = $O(|E| \log |V|)$

(2) Prove: 在 max-ST 上, 任何 s, t , 由 s 到 t 的 path 就是 s 到 t 的 widest path.

\Leftarrow 由定理

假設 p 是在 max-ST 上, 由 s 到 t 的路徑.

$\exists p'$ 不在 max-ST 上, p' 是 s 到 t 的路徑的 widest path

令 $e \in p$ path p 上的 minimum width,

$\therefore p'$ is wider path

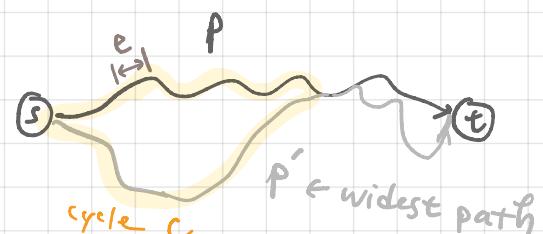
則 $\forall e' \in p', w(e') \geq (\text{minimum width on } p') > w(e)$

由於 p, p' 都是由 s 到 t 的路徑, 由 p, p' 的邊組成的

圖中的每個邊都位在某個 cycle 中, 則包含 e 的 cycle C ,

已知 $w(e)$ 的 C cycle 中最大的邊, \oplus modified cycle

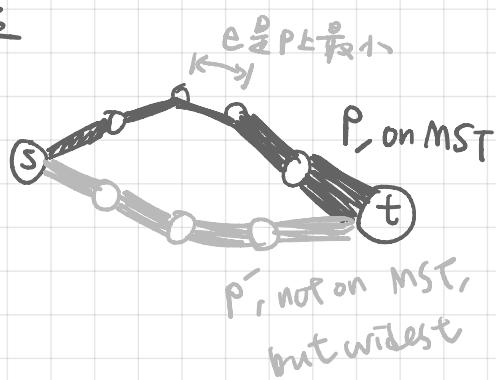
property, max-ST 不包含 e ($\rightarrow \Leftarrow$)



(*) modified cycle property. 將原本的 cycle property 套用至 max-ST 上,
且 maximum weight 改為 minimum weight
一樣能成立.

因此用 $O(E \log V)$ 找到的 max-ST 就是一樣

Pekora 想要的 set E'



(3) Prove: a road $(u, v) \in E$ is downward critical

\Leftrightarrow there is a shortest path from s to $u^{\text{or}}v$ ending at (u, v)

" \rightarrow " if a road (u, v) is downward critical, then there is a shortest path from s to $u^{\text{or}}v$ ending at (u, v)
反證法, if all shortest path from s to $u^{\text{or}}v$ doesn't end at (u, v)
then by decreasing $d(\text{edge } (u, v))$, if all the path containing (u, v)
still have bigger distance than original shortest path, then the
shortest path distance unchanged. \therefore edge (u, v) isn't downward
critical

" \leftarrow " if there is a shortest path from s to $u^{\text{or}}v$ ending at (u, v) ,
the shortest path has distance d , we know that $d < \text{any other}$
path distance
by decreasing $d(\text{edge } (u, v))$, we have new distance d' on
original path , and $d' < d < \text{any other path distance from } s \text{ to }$
 $u^{\text{or}}v$.
So the original path is still the shortest path and with
 \therefore road (u, v) is downward critical.

(4) <claim> a road (u, v) is upward critical \Leftrightarrow
all shortest path from S to $u^{or}v$ ends at (u, v)

" \rightarrow " if a road (u, v) is upward critical, then

all shortest path from S to $u^{or}v$ ends at (u, v)

<pf> $\exists i \nexists v_i$, if not, \exists some shortest path p from S to $u^{or}v$

doesn't end at (u, v) , then by increasing $d(u, v)$,
we can still have the shortest path p with unchanged
shortest distance as shortest path ($\rightarrow \Leftarrow$)

" \leftarrow " if all shortest path from S to $u^{or}v$ ends at (u, v)

then a road (u, v) is upward critical

<pf> Suppose the original shortest path have distance d , $d > \text{all distance of other path from } S \text{ to } u^{(or)v}$
by increasing $d(u, v)$, all of the original shortest path now have bigger distance d' , $d' > d$

\rightarrow if there exists other path that is not the original shortest path, but have distance $d'' < d'$, the shortest path distance would become d''

\rightarrow if not, the shortest path distance is d'

both d'' and $d' > d$, $\therefore (u, v)$ is upward critical

(5) - modify Dijkstra algorithm to find "all of the edges that are the last edge in some shortest path". (ref to 上課授業)

每個 node 記錄的 parent (u, v) 改為一個 vector,

在 Relax(u, v, w) 時, 若 $v.d = u.d + w(u, v)$, $v.v$ 新增 u

若 $v.d > u.d + w(u, v)$, $v.v = \{u\}$; $v.d = u.d + w(u, v)$

- 當執行完 Dijkstra 後, 所有 node u 和其 node 的 parent u, v

形成的 edge 都是 downward critical road.

若該 node u 的 parent u, v 只有一個, 則 u 和 u, v 形成的 edge 是 upward critical road

- Time complexity: 因為 G 沒重邊, 當執行 Dijkstra 可以用 min-priority queue

implement, 複雜度為 $O(E \log E + V) = O(E \log V)$ ($\because V-1 \leq E \leq V^2$)

最後輸出所有 upward / downward critical road 的複雜度為 $O(E)$

<pf> (brief explanation)

正確性, 在 Dijkstra 中, 已經證明 greedy choice: 每次選

$v.d$ 最小的加進 R 中, 可以確保當前 v, v node - 路往 parent trace 的路徑是最短路徑. 因為在 relax 時, 所有邊都會 relax, if w 最終只有 v, v 的值

由 path 的 parent 都會被納入 parent, 可以保證最終 $\forall u \in v, v$, $s^{(s, u)}$

$v.d = u.d + w(u, v)$, 又由 dijkstra $v.d = s(s, v)$ \therefore 以每個之內的 v, v - 路往 parent trace 都是最小路徑.

16) 題目要求：找 $\sum_{i=1}^n k(v_i)$ $> K$ 的 cycle,

即，在 cycle 內， $\sum_{i=1}^n k(v_i) > K^* \sum_{i=1}^n d(v_i, v_{i+1})$

$$\Rightarrow \frac{\sum_{i=1}^n k(v_i)}{K} > \sum_{i=1}^n d(v_i, v_{i+1})$$

$$\Rightarrow \sum_{i=1}^n d(v_i, v_{i+1}) - \frac{\sum_{i=1}^n k(v_i)}{K} < 0$$

重新定義任意邊 (u, v) 的 weight 为 $w(u, v) = d(u, v) - \frac{k(v)}{K}$

則對於 G 上任一 cycle C , 由 $v_1, v_2, \dots, v_n, v_{n+1} = v_1$ 且有：

C 的 weight 的加總為：

$$\sum_{i=1}^n w(v_i, v_{i+1})$$

$$= d(v_1, v_2) - \frac{k(v_2)}{K} + d(v_2, v_3) - \frac{k(v_3)}{K} + \dots + d(v_n, v_{n+1}) - \frac{k(v_{n+1})}{K}$$

$$= \sum_{i=1}^n d(v_i, v_{i+1}) - \frac{\sum_{i=1}^n k(v_i)}{K}$$

因此对于重新定義 weight 的圖 G 及 Bellman-ford algorithm,

課堂上已證明若存在負環（某邊 weight 的加總 < 0 ），則 return false.

所以當 Bellman-ford algorithm return false, 表不存在某 $v \in \{v_1, v_2, \dots, v_n\}$

使 $\sum_{i=1}^n d(v_i, v_{i+1}) - \frac{\sum_{i=1}^n k(v_i)}{K} < 0$ ← 這不是 existing cycle.

Time complexity the same as Bellman-ford algorithm = $O(VE)$