CPCFI

Lecture: MINIMUM SPANNING TREE II

Unit: 6

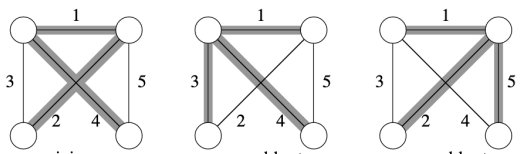
Instructor: NÉSTOR

I. SECOND BEST SPANNING TREE

↳ SPANNING TREE WITH THE SECOND MINIMUM WEIGHT SUM OF ALL THE EDGES

- T : MINIMUM SPANNING TREE OF GRAPH G
- T' : SECOND BEST SPANNING TREE

a) T IS UNIQUE WHILE T' IS NOT



– PROF. SHANG-HUA TENG'S HW 2 – SOLUTIONS – A

b) [ANY T' MUST HAVE AT LEAST ONE EDGE THAT IS NOT IN T]

↳ PROOF IN PROF. SHANG-HUA TENG'S HW 2 – SOLUTIONS – A :

b. Since any spanning tree has exactly $|V| - 1$ edges, any second-best minimum spanning tree must have at least one edge that is not in the (best) minimum spanning tree. If a second-best minimum spanning tree has exactly one edge, say (x, y), that is not in the minimum spanning tree, then it has the same set of edges as the minimum spanning tree, except that (x, y) replaces some edge, say (u, v), of the minimum spanning tree. In this case, $T' = T - \{(u, v)\} \cup \{(x, y)\}$, as we wished to show.

Thus, all we need to show is that by replacing two or more edges of the minimum spanning tree, we cannot obtain a second-best minimum spanning tree. Let T be the minimum spanning tree of G, and suppose that there exists a second-best minimum spanning tree T' that differs from T by two or more edges. There are at least two edges in $T - T'$, and let (u, v) be the edge in $T - T'$ with minimum weight. If we were to add (u, v) to T', we would get a cycle c. This cycle contains some edge (x, y) in $T' - T$ (since otherwise, T would contain a cycle).

We claim that $w(x, y) > w(u, v)$. We prove this claim by contradiction, so let us assume that $w(x, y) < w(u, v)$. (Recall the assumption that edge weights are distinct, so that we do not have to concern ourselves with $w(x, y) = w(u, v)$.) If we add (x, y) to T, we get a cycle c', which contains some edge (u', v') in $T - T'$ (since otherwise, T' would contain a cycle). Therefore, the set of edges $T'' = T - \{(u', v')\} \cup \{(x, y)\}$ forms a spanning tree, and we must also have $w(u', v') < w(x, y)$, since otherwise T'' would be a spanning tree with weight less than $w(T)$. Thus, $w(u', v') < w(x, y) < w(u, v)$, which contradicts our choice of (u, v) as the edge in $T - T'$ of minimum weight.

Since the edges (u, v) and (x, y) would be on a common cycle c if we were to add (u, v) to T', the set of edges $T' - \{(x, y)\} \cup \{(u, v)\}$ is a spanning tree, and its weight is less than $w(T')$. Moreover, it differs from T (because it differs from T' by only one edge). Thus, we have formed a spanning tree whose weight is less than $w(T')$ but is not T. Hence, T' was not a second-best minimum spanning tree.

TO FORM T' WE NEED TO FIND AN EDGE $(x,y) \notin T$ AND REPLACE IT WITH AN EDGE $(u,v) \in T$ SUCH THAT T' IS A SPANNING TREE AND THAT $(x,y) - (u,v)$ IS MINIMUM

[TWO APPROACHES]

1) KRUSKAL'S ALGORITHM ⇒ FIND T, REMOVE ONE EDGE AT A TIME AND TRY TO REPLACE IT WITH OTHER EDGE

•) FIND MST (T) USING KRUSKAL'S ALGORITHM
••) FOR EACH EDGE IN T, EXCLUDE IT FROM EDGES LIST
•••) FIND A NEW MST WITH REMAINING EDGES
••••) SELECT THE MST WITH LOWEST WEIGHT

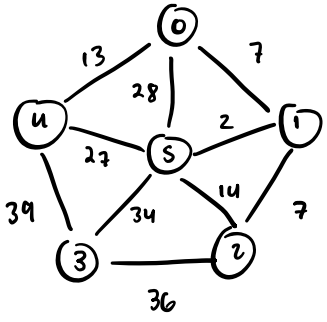
} O(VE) TIME

2) MODELING AS A LOWEST COMMON ANCESTOR PROBLEM ⇒ NOW, WE TRY TO ADD EVERY EDGE THAT IS NOT ALREADY IN THE MST

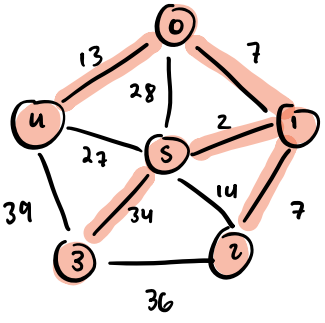
•) FIND MST USING KRUSKAL
••) ∀ (x,y) ∉ MST :
- ADD (x,y) TO MST
- FIND (u,v) WITH MAX. WEIGHT IN THE MST ((u,v) ≠ (x,y))
- REMOVE (u,v) TEMPORARILY
- FIND ∆ = WEIGHT (x,y) − WEIGHT (u,v)

↳ T' WILL BE THE SPANNING TREE WITH SMALLEST ∆

O(E log V) IF WE USE LOWEST COMMON ANCESTOR (LCA)

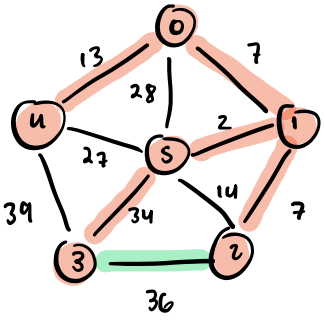


MST (USING KRUSKAL)



- NOW, WE HAVE TO FIND AN EDGE (x,y) THAT MINIMIZES ∆

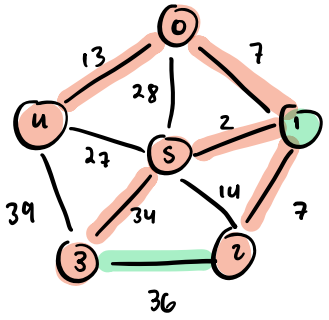
SUPPOSE WE PICK (3,2) AND ADD IT TO THE MST



THIS WILL FORM A CYCLE (3 − 5 − 1 − 2 − 3)

- NOW, WE HAVE TO FIND THE MAX. WEIGHT EDGE IN MST ≠ (3,2)

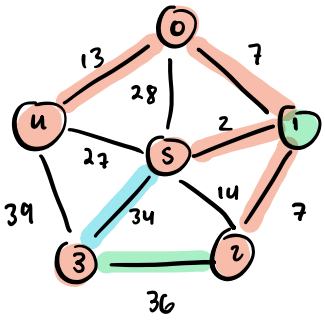
WE FIND THE LOWEST COMMON ANCESTOR OF EDGE (3,2)



LCA (3,2) = 1

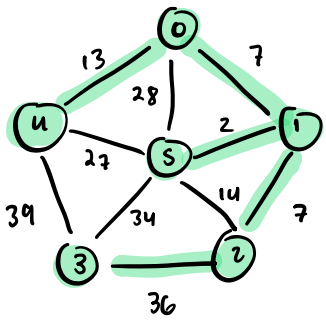
[NOW, WE COMPUTE THE MAX. WEIGHT EDGE ON THE PATHS FROM 2 → 1 AND 3 → 1]

LOWEST COMMON ANCESTOR



(3,s) IS THE MAX. WEIGHT EDGE

FINALLY, WE REMOVE (3,s) AND FIND δ



δ = WEIGHT (3,2) − WEIGHT (3,s)
= 36 − 34 = 2

II. KIRCHHOFF'S THEOREM

↳ USED TO FIND THE NUMBER OF DIFFERENT SPANNING TREES

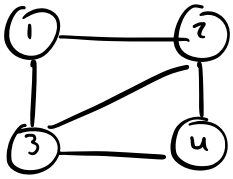
KIRCHHOFF'S THEOREM: THE NUMBER OF SPANNING TREES IS EQUAL TO ANY COFACTOR OF THE LAPLACIAN MATRIX OF G.

COFACTOR MATRIX IS THE MATRIX CREATED FROM THE DETERMINANTS OF THE MATRICES NOT PART OF A GIVEN COLUMN AND ROW:

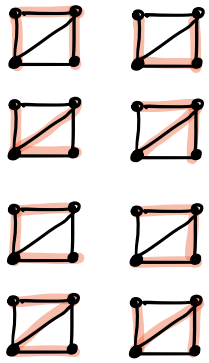
$$\operatorname{cof}\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} \begin{vmatrix} e & f \\ h & i \end{vmatrix} & -\begin{vmatrix} d & f \\ g & i \end{vmatrix} & \begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ -\begin{vmatrix} b & c \\ h & i \end{vmatrix} & \begin{vmatrix} a & c \\ g & i \end{vmatrix} & -\begin{vmatrix} a & b \\ g & h \end{vmatrix} \\ \begin{vmatrix} b & c \\ e & f \end{vmatrix} & -\begin{vmatrix} a & c \\ d & f \end{vmatrix} & \begin{vmatrix} a & b \\ d & e \end{vmatrix} \end{bmatrix}$$

$L = D - A$

- A → ADJACENCY MATRIX
- D → DEGREE MATRIX (DIAGONAL MATRIX WITH VERTEX DEGREES ON THE DIAGONAL)



$\Rightarrow$ SPANNING TREES : (8)



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \quad \begin{matrix} 1 & 2 & 3 & 4 \end{matrix}$$

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$L = D - A = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

DELETE THE ROW AND COLUMN 1 $\downarrow$

$$L^* = \begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

COFACTOR IS (1,1)

$\rightarrow$ DETERMINANT

$$|L^*| = 3(6-1) - (-1)(-2-1) + (-1)(1-(-3))$$

$$= 8 - 3 + 3$$

$$= \boxed{8} \text{ SPANNING TREES}$$

$\Downarrow$

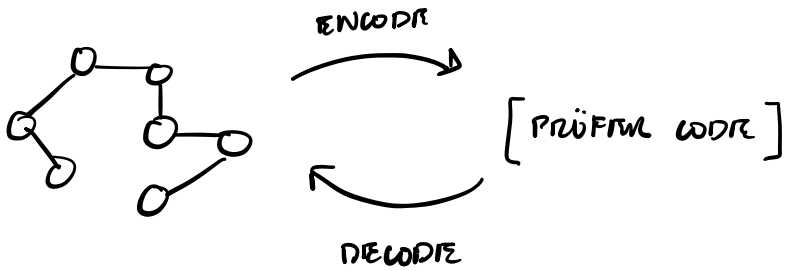
WE CAN USE GAUSSIAN METHOD $O(n^3)$ FOR FINDING IT

III PRÜFER CODE

- P.C. IS A WAY OF ENCODING A LABELLED TREE INTO A SEQUENCE OF NUMBERS IN A UNIQUE WAY
  - $\downarrow$ CONTAINS n VERTICES
  - $\hookrightarrow$ CONTAINS $n-2$ ELEMENTS $\in [0, n-1]$

- PROPERTIES :
  - SINCE $|P.C| = n-2$, 2 VERTICES WERE LEFT OUT FROM THE P.C., ONE OF IT IS THE HIGHEST VERTEX AND NOTHING CAN BE SAID FROM THE OTHER
  - EVERY VERTEX APPEARS IN THE P.C. ITS DEGREE MINUS ONE TIMES

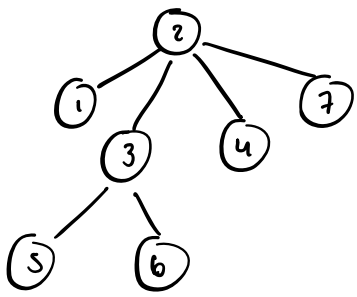
SIMPLE ILLUSTRATION :



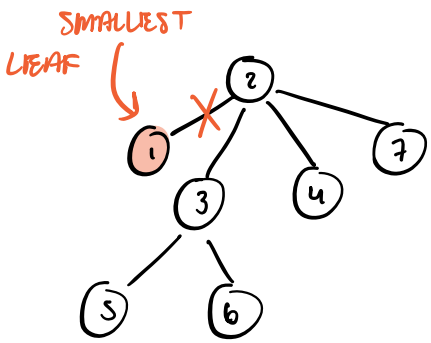
[STEPS TO BUID THE P.C.]

1. REPEAT THE PROCESS n-2 TIMES
2. SELECT THE LEAF WITH SMALLEST NUMBER
3. REMOVE THAT LEAF
4. ADD THE LEAF's PARENT TO THE P.C.

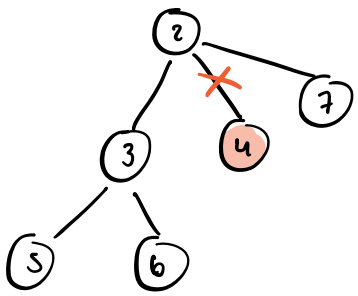
EXAMPLE : (BUILDING)



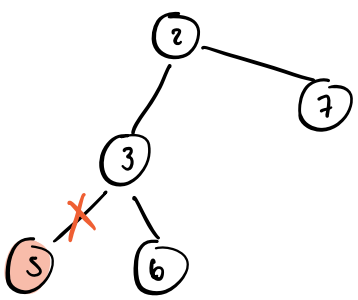
P.C. = { }



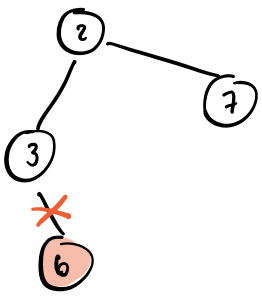
P.C. = 2



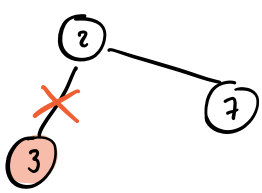
P.C. = 2 2



P.C. = 2 2 3



P.C. = 2 2 3 3



P.C. = 2 2 3 3 2

EXAMPLE: (PRÜFER TO NATION)

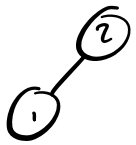
P.C. = 2 2 3 3 2

$\begin{bmatrix} n-2 = 5 \\ n = 7 \end{bmatrix}$

THE SMALLEST LEAF $\in [1, n]$ THAT IS NOT IN THE P.C. IS CONNECTED TO NODE (2)

NODE (1)

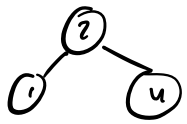
$\therefore$



P.C = 2 2 3 3 2

NEXT SMALLEST LEAF $\notin$ P.C. IS NODE (4)

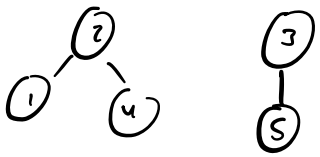
$\therefore$



P.C. = 2 2 3 3 2

NEXT SMALLEST LEAF $\notin$ P.C. IS NODE (5)

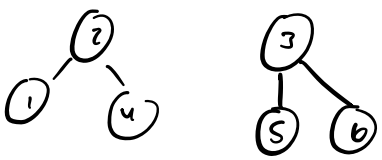
$\therefore$



P.C. = 2 2 3 3 2

NEXT SMALLEST LEAF $\notin$ P.C. IS NODE (6)

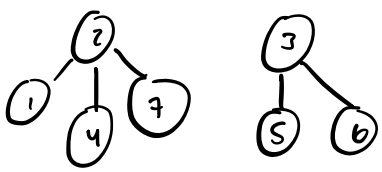
$\therefore$



P.C. = 2 2 3 3 2

NEXT SMALLEST LEAF $\notin$ P.C. IS NODE (7)

$\therefore$



FINALLY, WE CONNECT THE TWO VERTICES THAT DIDN'T GET REMOVED FROM THE P.C.

$\therefore$