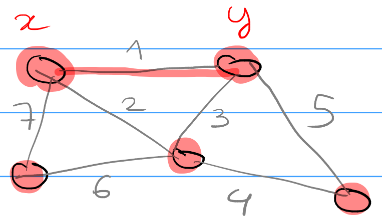
Exercice 1 : algorithme de Kruskal

1.1) – T est une forêt : aucun cycle n'est jamais créé (par construction)



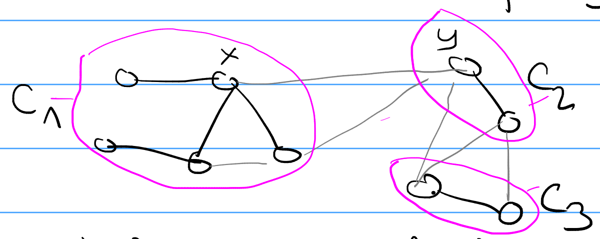
$H = \emptyset$
$H = \{x, y\}$

– T est couvrant.

Par l'absurde supposons que T ne contienne pas le sommet $u \in S$.
Par construct° l'algo a examiné les arêtes incidentes à u. Il aurait dû la 1ère de ces arêtes car elle n'aurait pas créé de cycle → T ne peut pas être obtenu par l'algo de Kruskal.

– T ne contient qu'une seule CC

Par l'absurde. Supposons qu'il y ait au – 2 CC, dont une appelée C_1.

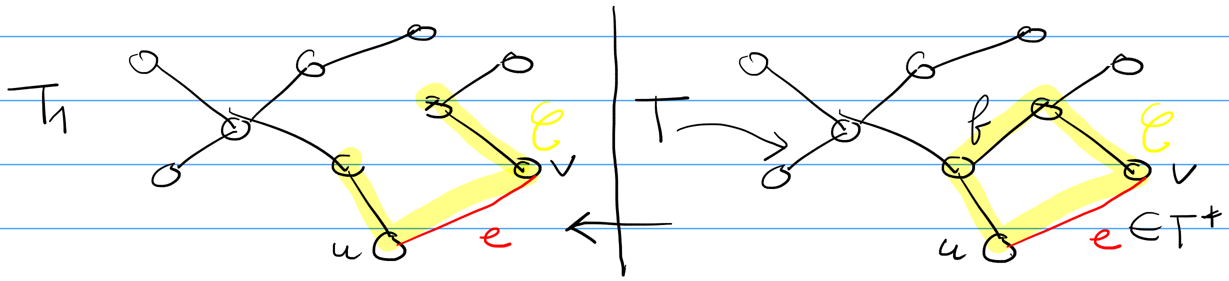


L'algo a examiné toutes les arêtes dont celles du co-cycle de C_1 ($\exists$ au – 1 arête dans ce cocycle car G est connexe et C_1 ne couvre pas tout G).

Quand l'algo examine la 1ère arête du co-cycle de C_1, celle-ci ne crée pas de cycle et pourtant elle n'est pas sélectionnée. Contradiction → T ne peut pas être obtenu par l'algo. de Kruskal.

1.2) Soit T^* un ACCM, T_1 l'arbre retourné à l'étape précédente. ~~par l'algo de Kruskal~~.
Soit e_1 l'arête de + petit coût dans $T^* \setminus T_1$

1.3)



– T_1 est un AC → $T \cup \{e\}$ est un arbre auquel on a ajouté 1 arête : on a donc 1 cycle $\mathcal{C}$. En retirant l'arête f de $\mathcal{C}$ on obtient un arbre.
Cet arbre est un AC car il contient tous les sommets de G.

– T_1 a une arête en commun avec T^* de plus que T (car $e \in T^*$ et $f \notin T^*$)

– $\text{coût}(T_1) = \text{coût}(T) - \text{coût}(f) + \text{coût}(e)$
$\geq \text{coût}(T)$ car $\text{coût}(f) \leq \text{coût}(e)$

1.4) Ce qu'on a fait :

$$T \xrightarrow{\text{algo Kruskal}} T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow \cdots \rightarrow T_k = T^*$$

a une arête de + en commun avec T^*
$coût(T_1) \geq coût(T)$.

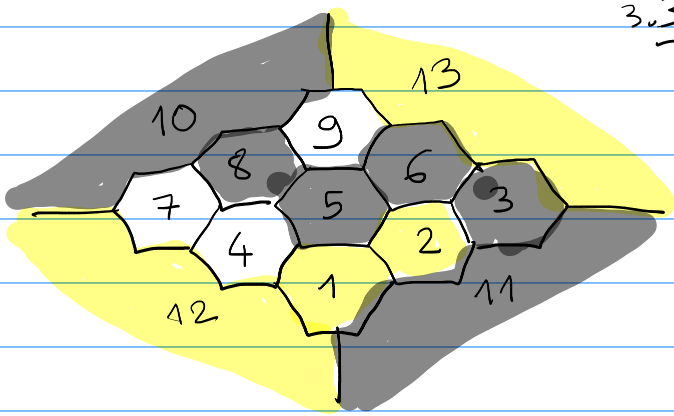
On réitère ce processus jusqu'à obtenir un arbre couvrant $T_k = T^*$.

$$coût(T) \underset{=}{\leq} coût(T_1) \underset{=}{\leq} coût(T_2) \underset{=}{\leq} \cdots \underset{=}{\leq} coût(T_k) = T^*$$

Comme T^* est un ACCM, on a $coût(T) = coût(T^*)$

Exercice 3

3.2) Partition initiale :
$(\{1\}, \{2\}, \{3\}, \ldots, \{12\}, \{13\})$.



3.3) Union(5,8) et
Union(5,6)

3.4) Find(10) = Find(11) ?
Si oui le joueur noir a gagné.

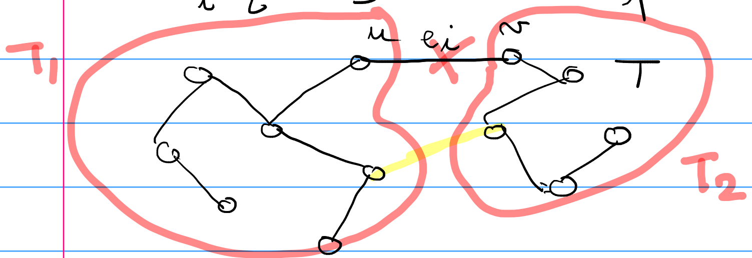
3.1) Elements $\{1,2,3,\ldots,13\}$
Un ensemble d'elts forme une classe d'équivalence en cours de partie si ils sont tous de la même couleur (noir et gris) et s'ils sont tous connectés ensemble par une chaîne de pions noirs (ou gris)

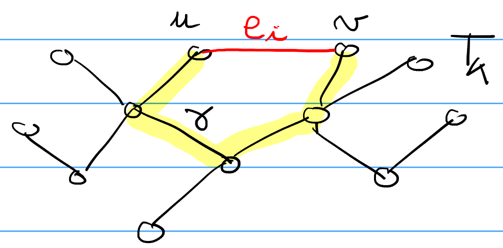
Exercice 2

Hypothèse : $c(e_1) < c(e_2) < \cdots < c(e_m)$. Par l'absurde :

1) Soit T_K l'ACCM obtenu avec l'algo de Kruskal.
Soit T un autre ACCM ($T \neq T_K$).

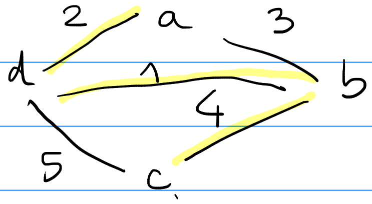
Soit $e_i = \{u,v\}$ l'arête de + petit coût appartenant à T et pas à T_K.





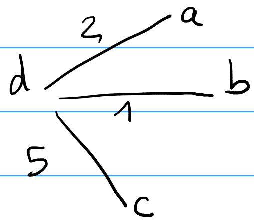
- Si l'algo de Kruskal n'a pas sélectionné $e_i = \{u,v\}$ c'est qu'il existait une chaîne γ entre u et v au moment de l'examen de e_i. Par construction, le coût de chaque arête de γ est $<$ au coût de e_i.
- En retirant e_i à T on obtient 2 arbres T_1 et T_2. On sait qu'il existe une arête e' de γ dans le co-cycle de T_1 (car $u \in T_1$ et $v \in T_2$). En remplaçant e_i par e' dans T, on obtient un AC et de coût $coût(T) - coût(e_i) + coût(e') < coût(T) \Rightarrow T$ n'est pas un ACCM. Contradiction.

2)

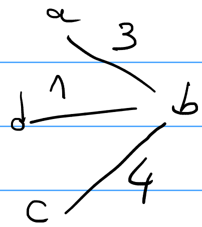


ACCM : Coût : 7

2ere meilleur arbre



coût = 8



coût = 8