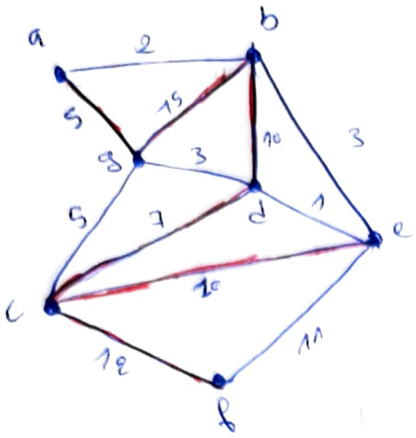
Algorithme de Prim

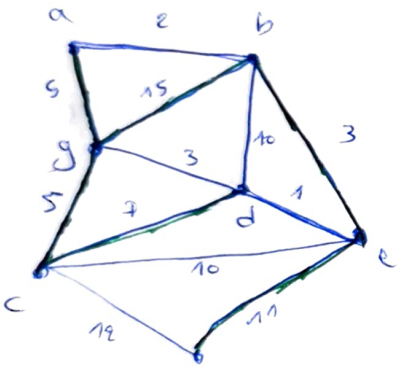
On a un graphe connexe avec la particularité d'être pondéré c-à-d qu'on va attribuer à chaque arête une pondération (un poids qui a une valeur positive).



- On veut extraire un arbre qui contient tous les sommets donc un arbre couvrant et il faut qu'il soit de plus petit poids.
- si on prend un exemple (les arêtes en rouge)
- faut vérifier si ça satisfait bien les conditions :
- c'est un arbre
- ça contient tous les sommets
- mais faut vérifier si c le poids min.

Poids (T) = 5+15+10+7+10+12 = 59

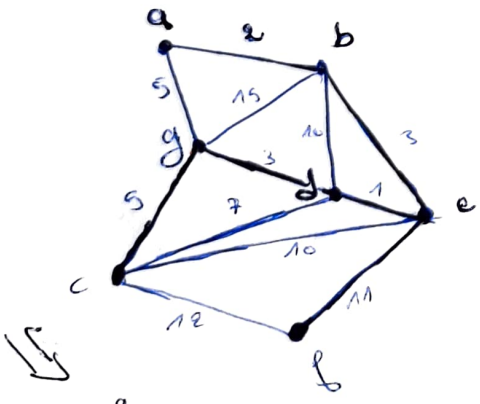
- Prenons une autre arbre (arêtes en vert).



- c'est un arbre
- tous les sommets
- Poids(T) = 5+15+5+7+3+11 = 46 < poids(T)

↳ donc l'arbre rouge n'était pas de poids optimal.
↳ faut vérifier si le poids de l'arbre vert est optimal.

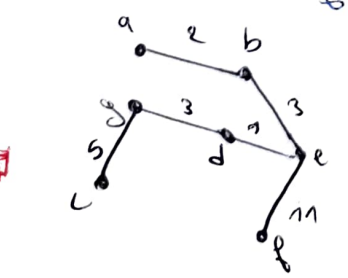
⟶ Description d'algo de prim (qui prend en entrée un graphe connexe pondéré et va construire un arbre couvrant de poids minimal)



1ère étape : on considère n'importe quel sommet de départ par exemple le sommet (d)

- à chaque étape on va ajouter à ce sommet un sommet et une arête pour agrandir l'arbre partiel (qui est que (d) pour l'instant). L'arête qui va être choisie c'est celle du poids minimal (dans notre cas c'est l'arête de e avec un poids de 1)

2ème étape : on va appliquer la même règle. on va regarder toutes les arêtes qui ont une extrémité dans l'arbre et une extrémité en l'arbre on a donc eb, db, dg, dc, ec et ef. on va ajouter celle qui est de poids minimal. Là on a 2 (eb et dg) avec 3 comme poids on choisit n'importe laquelle. prenons exemple eb.



poids(T) = 2+3+1+3+5+11 = 25

3ème étape : même démarche. Les extrémités sont ba, bg, dg, dc, ec et ef.

Attention : l'arête entre de et b n'est pas une arête candidate puisqu'elle a ses deux extrémités dans l'arbre

4ème étape : pareil : extrémités ag, bg, dg, dc, ec, ef