Conception d'algorithmes

Algorithmes gloutons

V. Chevrier

Algorithmes gloutons

- Problème d'optimisation
  - Utiliser une optimalité locale en espérant une optimalité globale
- À chaque étape faire un choix qui semble le meilleur à ce moment
- Solution optimale pas forcément trouvée

- Exemples introductifs
  - Ordonnancement d'activités
  - Monnayeur
  - …

Exemple introductif

Ordonnancement d'activités

- Soit un ensemble S d'activités utilisant une ressource
- Chaque activité i possède une date de début s_i, une date de fin f_i
- Deux activités sont compatibles ssi $s_i >= f_j$ ou $s_j>=f_i$ (ie elles ne se recouvrent pas)
- Indiquer le plus grand ensemble d'activités mutuellement compatibles

données

i	1	2	3	4	5	6	7	8	9	10	11
si	1	3	0	5	3	5	6	8	8	2	12
fi	4	5	6	7	8	9	10	11	12	13	14

Proposition

- Hyp:
  - les activités sont triées par date de fin croissante
  - On les numérote selon leur rang
- Solution initiale A = {1} // activité qui se termine le plus tôt
- Choisir à chaque étape la première activité restante compatible avec la date de fin de la dernière ajouté à A

Proposition

- Hyp: les activités sont triées par date de fin croissante
- Solution initiale A = {1} // activité qui se termine le plus tôt
- Choisir à chaque étape la première activité restante compatible avec la date de fin de la dernière ajoutée à A
- Pseudo code
  - N = taille (S)
  - A = {1} ; j ← 1
  - Pour i de 2 à n
    - Si $s_i >= f_j$ alors A ← A U i; j ← i

Exemple

i	1	2	3	4	5	6	7	8	9	10	11
si	1	3	0	5	3	5	6	8	8	2	12
fi	4	5	6	7	8	9	10	11	12	13	14

Exécution de l'algorithme= …

[Cormen et al. 91]

Exemple

i	*1*	2	3	4	5	6	7	8	9	10	11
si	*1*	3	0	5	3	5	6	8	8	2	12
fi	*4*	5	6	7	8	9	10	11	12	13	14

Exécution de l'algorithme= …

[Cormen et al. 91]

Exemple

i	***1***	**~~2~~**	**~~3~~**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
si	*1*	~~3~~	~~0~~	5	3	5	6	8	8	2	12
fi	*4*	~~5~~	~~6~~	7	8	9	10	11	12	13	14

Exécution de l'algorithme= …

[Cormen et al. 91]

Exemple

i	***1***	~~**2**~~	~~**3**~~	**4**	~~**5**~~	~~**6**~~	~~**7**~~	**8**	~~**9**~~	~~**10**~~	**11**
si	*1*	~~3~~	~~0~~	5	~~3~~	~~5~~	~~6~~	8	~~8~~	~~2~~	12
fi	*4*	~~5~~	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	~~13~~	14

Exécution de l'algorithme= …

[Cormen et al. 91]

Caractéristique de la solution … ?

- Cout de construction
- Optimalité

Caractéristique de la solution … ?

- Cout de construction
  - n pour la construction
  - n log(n) pour la préparation des données (tri)

Caractéristique de la solution … ?

- Cout de construction
  - N pour la construction
  - N log(n) pour la préparation des données (tri)
- Optimalité
  - Ici oui
  - Explication
    - Soit S un ensemble d'activités trié par date de fin
      - 1 a la plus petite date de fin
    - Soit une solution optimale A, triée par date de fin
    - Soit k la 1ère activité
      - Si k= 1 ➔ A commence par le choix glouton
      - k≠1 ➔ montrons alors qu'une autre solution optimale existe et commence par 1

- Soit B = A-{k} U {1}
  - $f_1 <= f_k$ (par construction) => activités dans B disjointes
  - Taille(B) = taille(A)
  - B est aussi optimal

  ➔ *il existe donc toujours une autre solution optimale commençant par le choix glouton*
- Soit A' = A-{1} est solution optimale pour S'={i dans S tq $s_i >= f_1$}
  - B' solution de S' avec plus d'activités que A' ne peut exister
  - En effet sinon B' U {1} serait solution optimale de A (mais avec plus d'activités => contradiction avec l'optimalité de A)
- *Donc une fois le choix glouton fait, on se retrouve sur le même problème d'optimisation que celui d'origine*

Fin de l'explication

- Par induction sur le nombre de choix
  - Faire le choix glouton à chaque étape produit une solution optimale ☺

Pièce de monnaie et approche gloutonne

- 5, 20, 1, 50, 2, 200, 10, 100 somme =37
- ?

Pièce de monnaie et approche gloutonne

- *Rem: il faut « préparer » les données pour faire les choix à chaque étape*
  - *Trier les données*
  - *Ne prendre une pièce que si ça ne dépasse pas sinon chercher avec la valeur immédiatement inférieure*
- 200, 100, 50, 20, 10, 5, 2, 1 somme =37
- Exécution de l'algo ?

Pièce de monnaie et approche gloutonne

- *Rem: il faut « préparer » les données pour faire les choix à chaque étape*
  - *Chercher la valeur immédiatement inférieure*
  - *trier les données*
- 200, 100, 50, 20, 10, 5, 2, 1 somme =37
- Prendre la pièce qui réduit le plus somme à rendre
  - Prendre 20
  - Prendre 10
  - Prendre 5
  - Prendre 2
  - Gagné !!
- C'est le plus petit nombre de pièces possible ici
- Nombre d'opérations O(n)

Autre exemple (non optimal)

- Pièces = 1, 1, 1, 1, 1, 10, 10, 15 Somme = 20
- Exécution : ?

Éléments pour la construction d'algorithmes gloutons

- Choix glouton et propriété
  - Solution globalement optimale construite à partir de choix localement optimaux
  - Le choix local ne dépend que de l'état courant de la solution (pas du futur) => approche top-down
  - Pas de remise en cause des choix
- Sous-structure optimale
  - Un problème possède une sous-structure optimale si une solution optimale au problème contient les solutions optimales au sous-problèmes

Principe

- On part d'une première donnée (D les données *sous la forme qui va bien*)
- Suite (sans remise) en cause de choix

- A ← vide
- Tant que non vide (D)

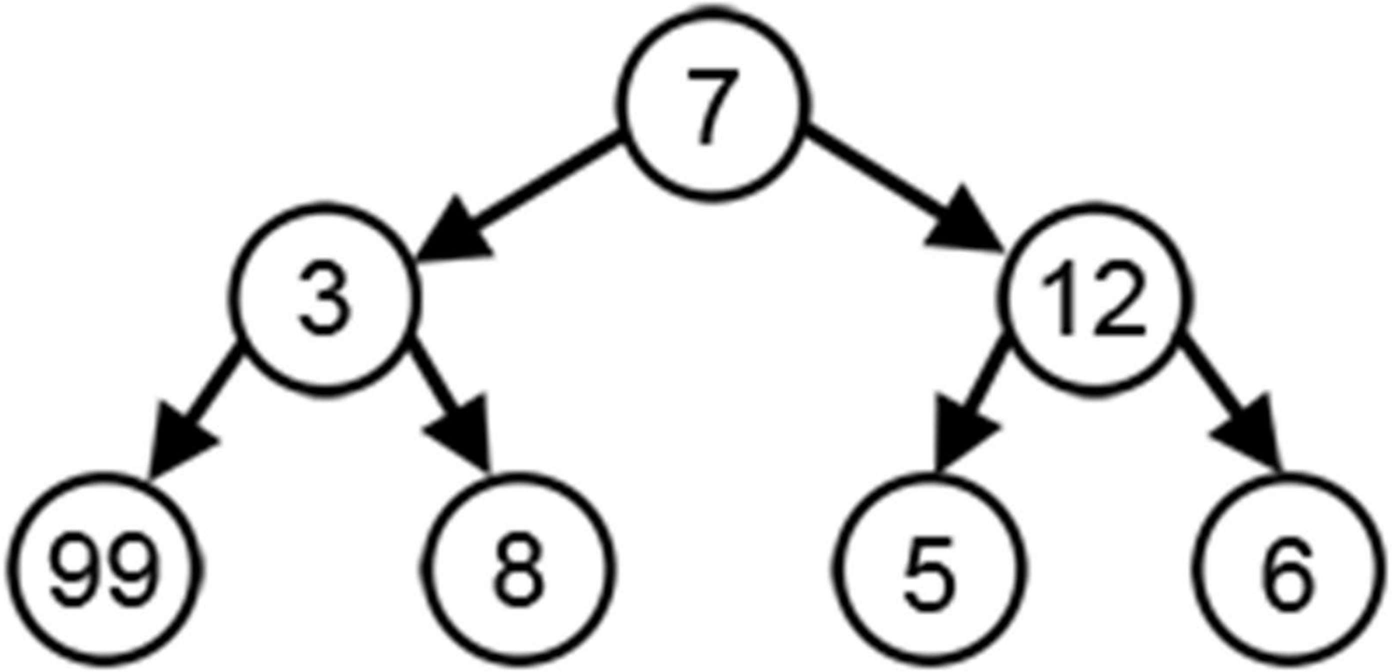
  choix← premier(D)

  Si compatible(choix)

  A ← A U {choix}

  D ←D-{choix}

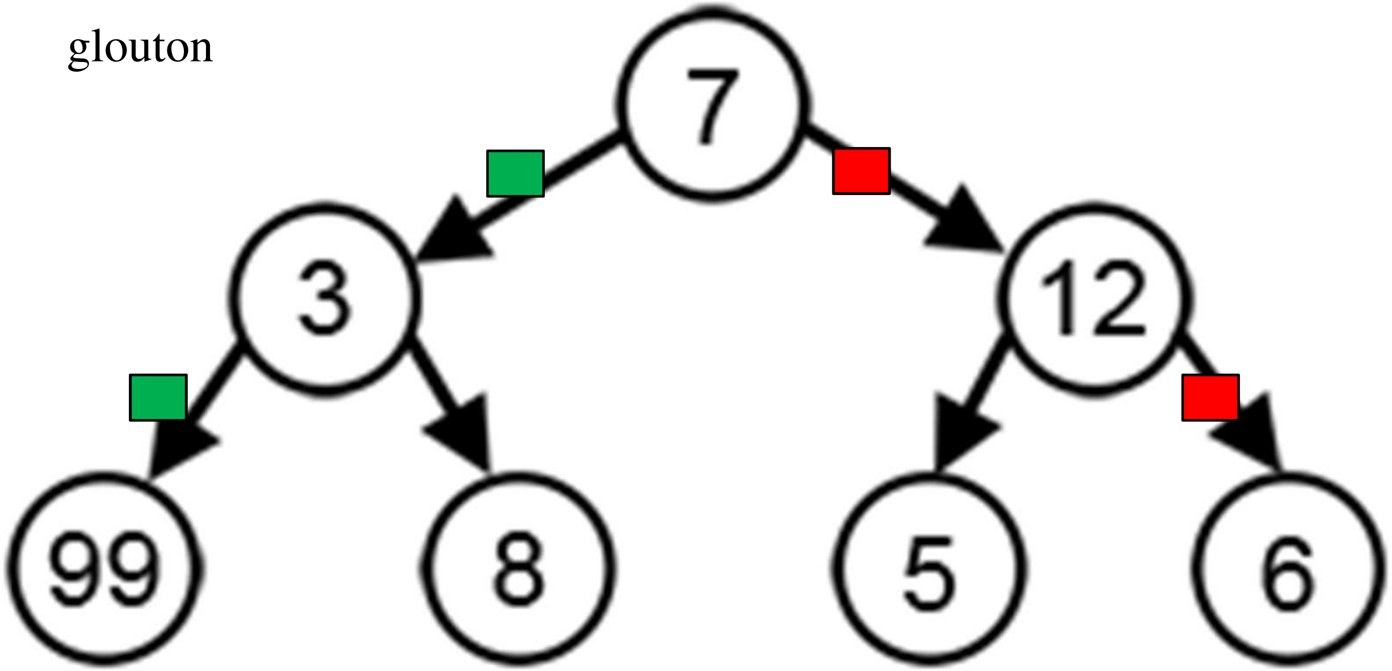
- Application (on maximise la somme à chaque niveau)



- Application (on maximise la somme à chaque niveau)

optimum

glouton



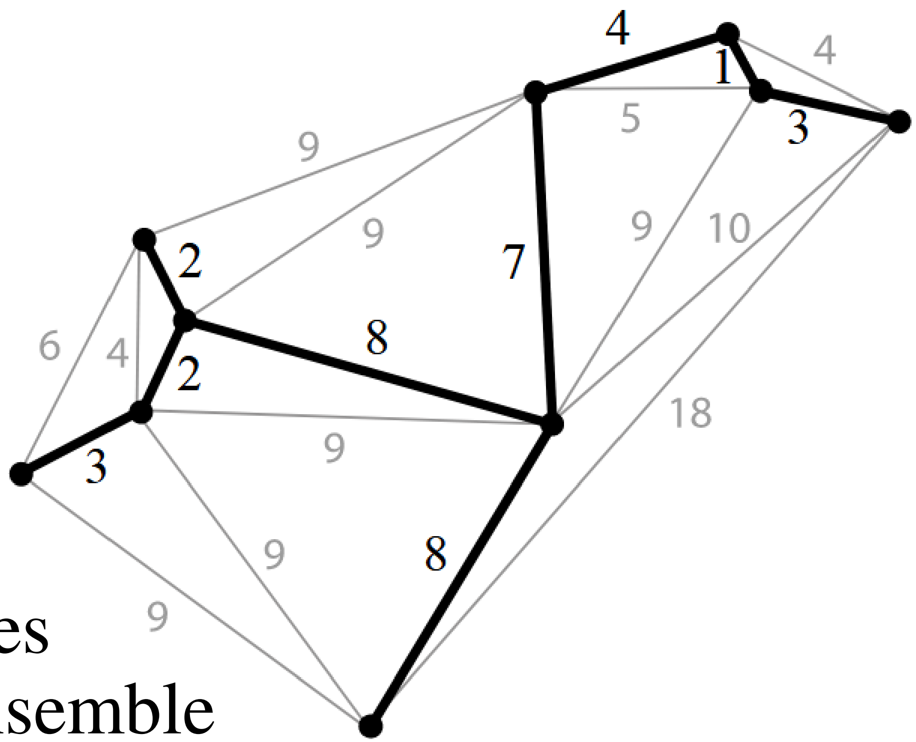
Graphe et algorithme glouton

- Arbre couvrant maximal
  - Kruskal
  - Prim
- Codage de Huffman

Arbre couvrant minimal

- graphe connexe valué et non orienté

(c) wikipedia



- Trouver une suite d'arêtes formant un arbre sur l'ensemble des sommets du graphe initial, et tel que la somme des poids de ces arêtes soit minimale

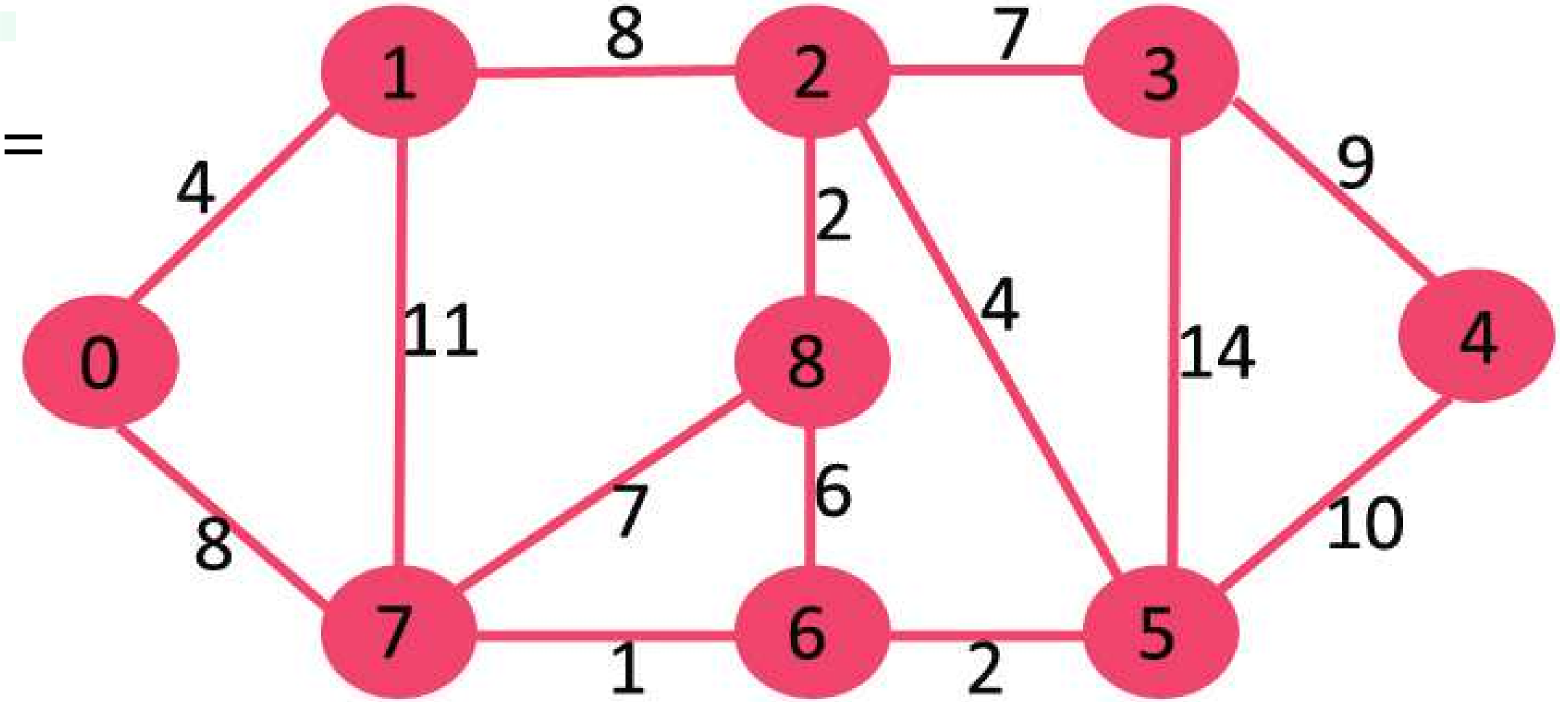
Principe : algorithme de Prim

- Choisir un sommet arbitraire
- Choisir l'arête incidente à ce sommet de poids minimal et on l'ajoute à l'arbre
- Choisir l'arête de poids minimum qui relie un sommet de l'arbre avec un sommet en dehors de l'arbre.

➔ Trier les arêtes selon leur cout

Algorithme de Prim

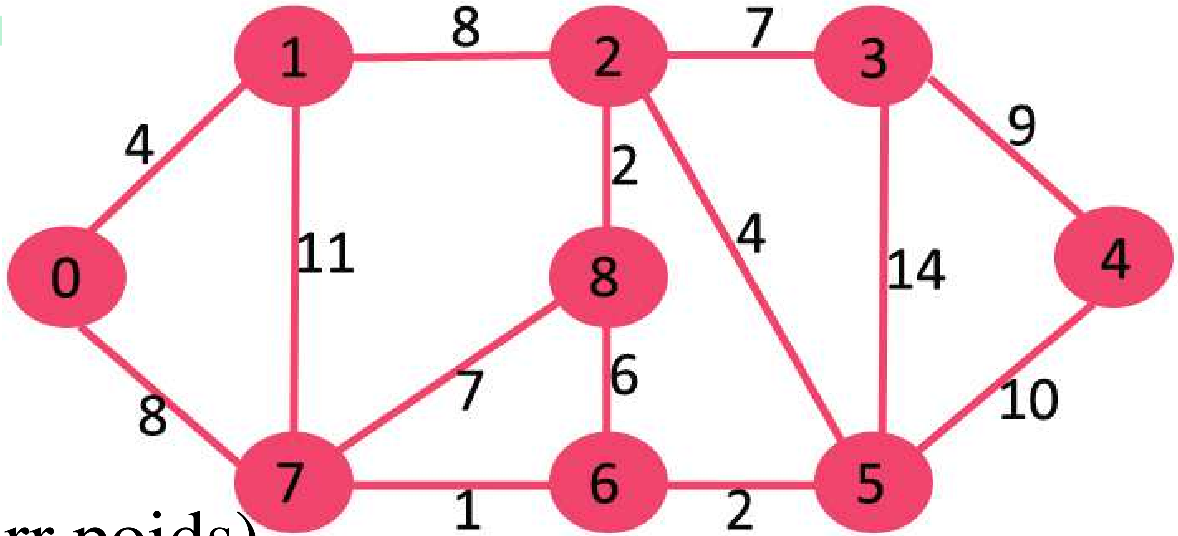
- Soit G =



- On commence par 0
- ACM ={0}

© geekforgeeks

Algorithme de Prim



- Arêtes triées: (dep,arr,poids)

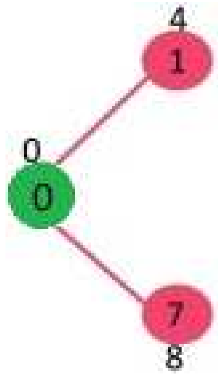
  (6,7,1)(5,6,2)(2,8,2)(0,4,4)(2,5,4)(6,8,6)(2,3,7)(7,8,7)
  (0,7,8)(1,2,8)(3,4,9)(5,4,10)(1,7,11)(3,5,14)

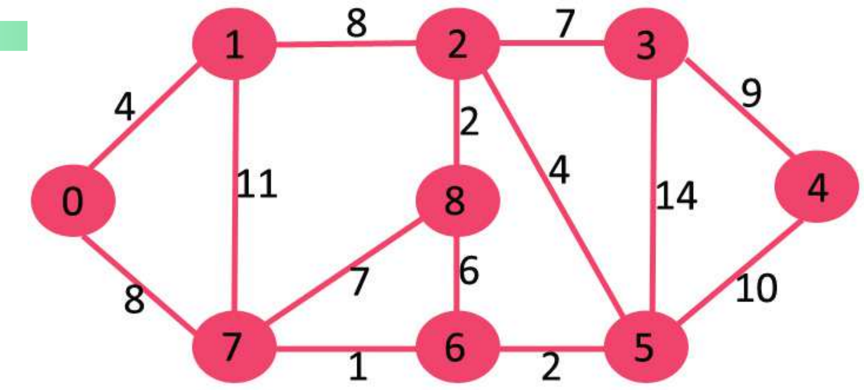
© geekforgeeks

Algorithme de Prim

- On commence par 0
- ACM ={0}
- Étape 1

  ACM={0,1}

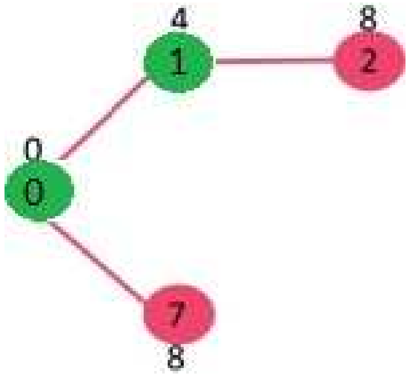




poids d'un sommet = valeur minimale de l'arête

- Etape 2

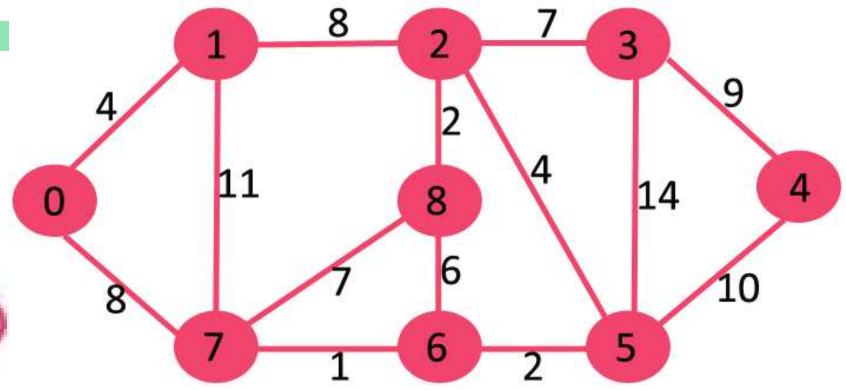
  ACM={0,1,*<u>7/2</u>*}

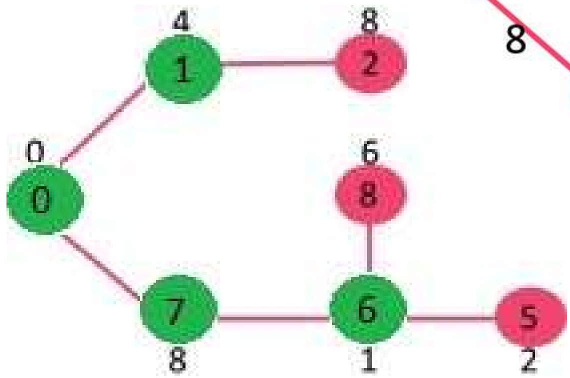


Algorithme de Prim

- On commence par 0
- ACM ={0}
- Étape 3

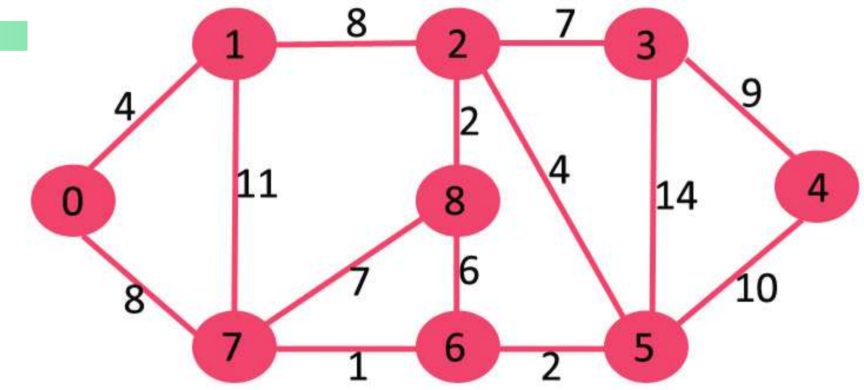
  ACM={0,1,7,6}





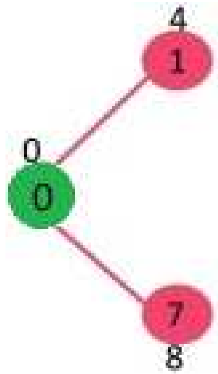
- etc

Algorithme de Prim



- On commence par 0
- ACM ={0}
- Étape 1

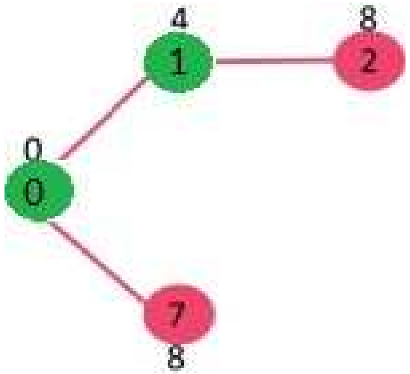
  ACM={0,1}
  avec poids=(0,4)



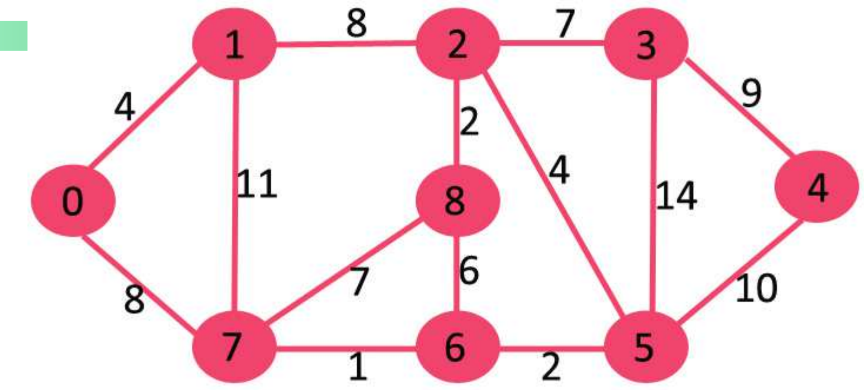
- Etape 2

  ACM={0,1,2}

  Et poids =(0,4,8)

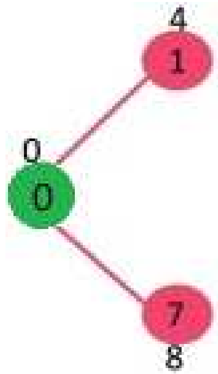


Algorithme de Prim



- On commence par 0
- ACM ={0}
- Étape 1

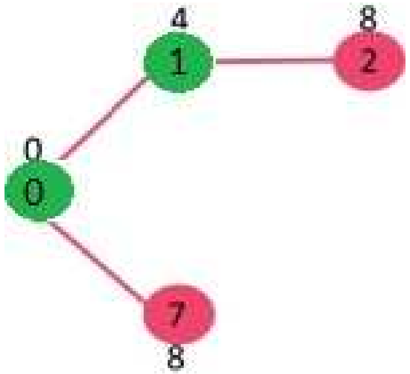
  ACM={0,1}
  avec poids=(0,4)



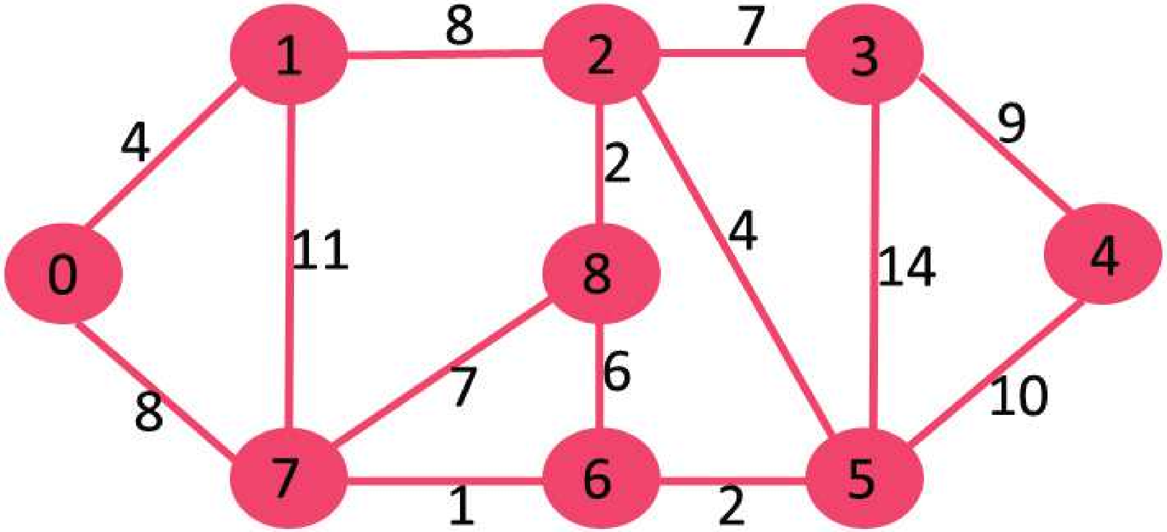
- Etape 2

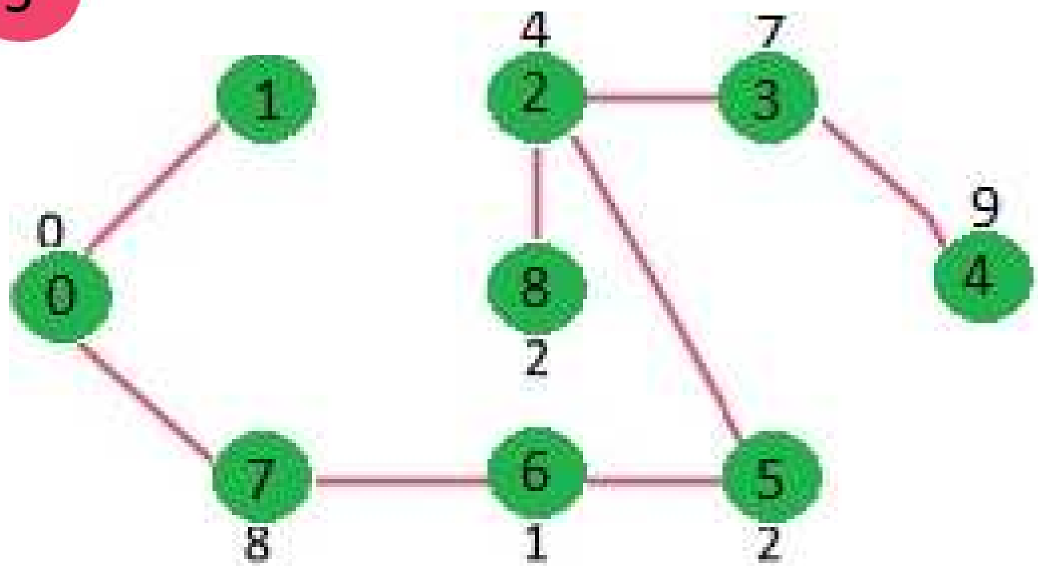
  ACM={0,1,2}

  Et poids =(0,4,8)



Algorithme de Prim





Codage de Huffman

- Codage optimal
- On dispose d'un texte (e.g.) et pour chaque lettre de sa fréquence d'apparition

	a	b	c	d	e	f
fréquence	45	12	24	8	5	6

Préliminaire

- Codage binaire
  - taille fixe
    - chaque caractère est codé avec un même nombre de bit
  - taille variable
    - les caractère les plus fréquents ont un code plus petit que les caractères rares.

	a	b	c	d	e	f
Code fixe	001	010	011	100	101	110
Code variable1	0	10	11	101	110	111
Code variable 2	0	101	100	111	1101	1100

Préliminaire

- Code à préfixe
  - tous les mots du code commence par le même préfixe et que aucun mot n'est préfixe d'un autre

	a	b	c	d	e	f
fréquence	45	12	24	8	5	6

	a	b	c	d	e	f
Code fixe	001	010	011	100	101	110
Code variable1	0	10	11	101	110	111
Code variable 2	0	101	100	111	1101	1100

N = 1000
Taille du code du texte dans chaque cas
Remarque ?

Préliminaire

- Comment code t-on un texte à l'aide d'un code à préfixe ?
- Comment décode t-on un texte codé ?
- Comment représenter le code pour pouvoir décoder facilement ?

Exemple

a	b	c	d	e	f
45	13	12	16	9	5
0	101	100	111	1101	1100