PLANS ET RAPPORTS DE TESTS

Plan de tests

Tests Fonctionnels : classe Node

- Function1 : bool addVoiz (Node&, Int)

  si addVoiz (noeud1,2)=vrai alors

  isVoiz (noeud1) doit valoir vrai

  finsi

  finTestFunction1 :

- Function2 : bool delVoiz(Node&)

  Si delvoiz (noeud1) alors

  isVoiz(noeud1) doit valoir faux

  finsi

  finTestFunction2

- Function3 : void afficheTable ()

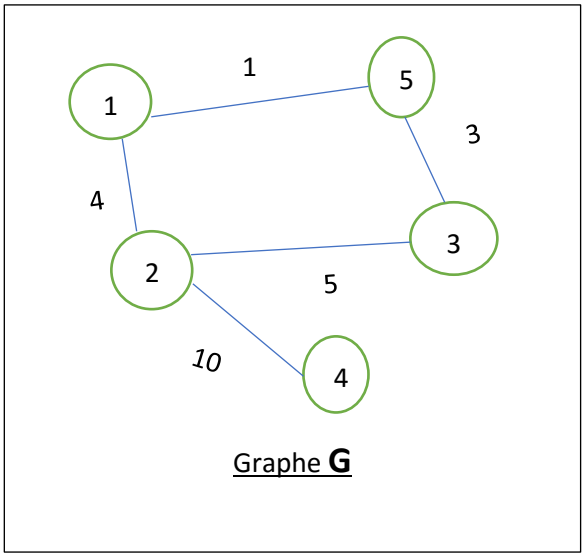
  Si le graph est G alors

  aTable de 1 est :

1	1	0
1	2	4
1	3	4
1	4	14
1	5	1

  Finsi

  finTestFunction3



Graphe **G**

- Function4 : void dikjstra()

  Si le graphe est G alors

  Dikjstra doit donner (voir annexe 2)

  Finsi

  finTestFunction4

Test sur classe graphe

- Function1 : Graphe( )

  A la fonction Graphe on donne les paramètres suivants

  Graphe (4, 3, 500)

Alors

Pour i ← 1 à 3 faire

Fichier.Exist(dossierCom/i.txt) est vrai

Finpour

FinTestFunction1

- Function2 : initialisation

Si nbrNode = i

Nodes.size () = i

linkTable.size () = i

linkTable[0].size () = i

finsi

finTestFunction2

ANNEXE 2

Nœud 1

1	2	3	4	5
0*	4	∞	∞	1
	4	4	14	1*
	4*	4	14	

Nœud 2

2	1	3	4	5
0*	4	5	10	∞
	4*	5	10	5
		5*	10	5

Nœud 3

3	1	2	4	5
0*	∞	5	∞	3
	4	5	15	3*
	4*	5	15	

Nœud 4

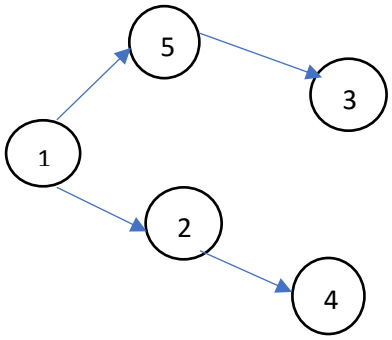
4	1	2	3	5
0*	∞	10	∞	∞
	14	10*	15	∞
	14*		15	15
			15*	15

Nœud 5

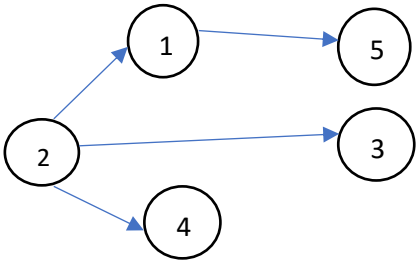
5	1	2	3	4
0*	1	∞	3	∞
	1*	5	3	∞
		5	3*	15
		5*		15

Les chemins les plus courts

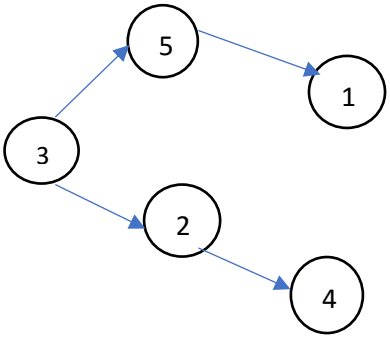
Nœud 1



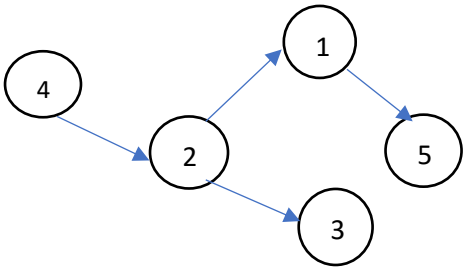
Nœud 2



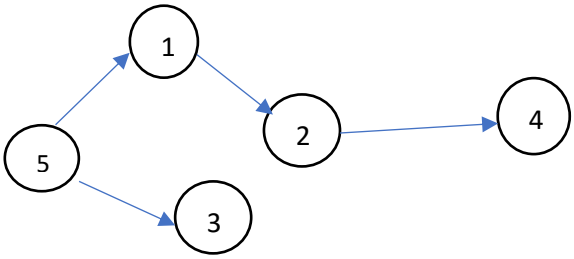
Nœud 3



Nœud 4

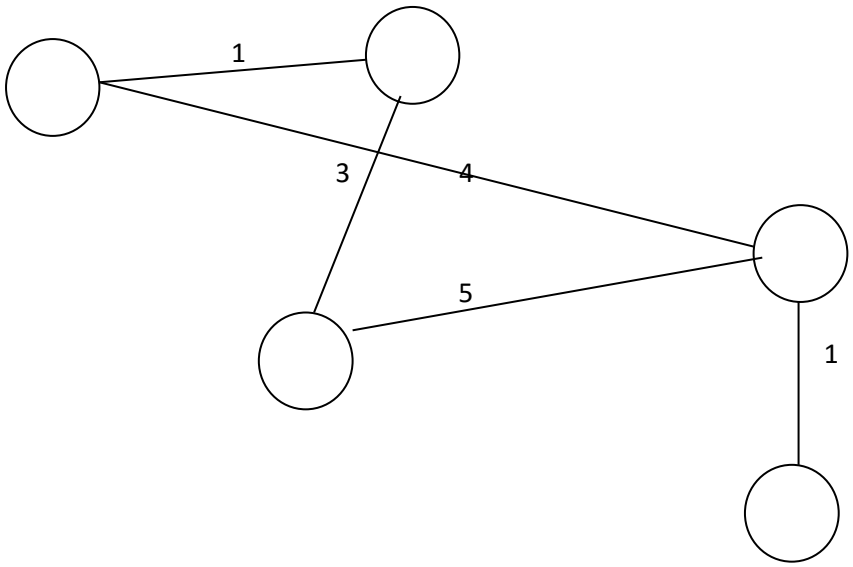


Nœud 5



RAPPORT DE TESTS FONCTIONNELS

Le test est appliqué sur le graphe suivant



Légende : G

Nous avons déclaré **graph,** une variable de type graphe qui est la présentation numérique de graphe G.

Note : dans la représentation dans le graphe, le nœud 1 est représenté par 0.Le nœud 1 sera donc le nœud i-1.

Test sur la classe Node

Fonction 1 :bool addvoiz(node&, int)

Tableau des resultats :testé sur le nœud 1-1

N° test	Valeur en entrée	Valeur isVoiz() attendu	Valeur de isVoiz() obtenu	Résultats attendus	Résultats obtenu
1	graph.nodes[2-1]	Vrai	Vrai	Vrai	Vrai
2	graph.nodes[4-1]	Vrai	Vrai	Vrai	Vrai
3	graph.node[5-1]	Vrai	Vrai	Vrai	Vrai

Fonction 2 :bool delVoiz(node&)

Tableau des résultats :test sur le nœud 1-1

N° test	Valeur en entrée	Valeur de isVoiz() attendu	Valeur de isVoiz() obtenu	Résultat attendu	Résultat obtenu
1	graph.nodes[4-1]	Faux	Faux	Faux	Faux
2	graph.nodes[3-1]	Faux	Faux	Faux	Faux
3	graph.nodes[1-1]	Faux	Faux	Faux	Faux
4	graph.nodes[2-1]	Faux	Faux	Faux	Faux

Fonction 3 : void affichtable()

Tableau des résultats : testé sur le nœud

Résultats attendus			Résultats obtenus		
1	1	0	1	1	0
1	2	4	1	2	4
1	3	4	1	3	4
1	4	14	1	4	14
1	5	1	1	5	1

Fonction 4 : void djikstra

Nœuds	Résultats attendus	Résultats obtenus
1-1	1 2 1 2 4 1 5 1 5 3	1 2 1 2 4 1 5 1 5 3
2-1	2 1 2 3 2 4 2 1 5	2 1 2 3 2 4 2 1 5
3-1	3 5 1 3 2 3 2 4 3 5	3 5 1 3 2 3 2 4 3 5
4-1	4 2 1 4 2 4 2 3 4 2 1 5	4 2 1 4 2 4 2 3 4 2 1 5

5-1	5 1 5 1 2 5 3 5 1 2 4	5 1 5 1 2 5 3 5 1 2 4

Test sur la classe Graphe

Fonction 1 : Graphe(int, int, int, std ::string)

Valeur entrée	Eléments testés	Résultat de l'existence attendue	Résultats obtenus
(4, 3, 500)	dossierCom/0.txt	Vrai	Vrai
	dossierCom/1.txt	Vrai	Vrai
	dossierCom/2.txt	Vrai	Vrai
	dossierCom/3.txt	Vrai	Vrai
	dossierCom/4.txt	Vrai	Vrai

Fonction 2 :

N°	Eléments testés	Valeurs en entrée	Valeurs à rendre	Valeurs obtenues
1	nodes.size()	4	4	4
	linkTable.size()	4	4	4
	linkTable[0].size()	4	4	4
2	nodes.size()	6	6	6
	linkTable.size()	6	6	6
	linkTable[0].size()	6	6	6