Analyse av metoder

Klasse:	Metode:	O notasjon:
LoadNumbers	getFirst	O(1)
	getInt	O(1)
	getSize	O(1)
	loadFile	O(n) i henhold til lengden av tekst fil
Task1	getInput	O(n) i henhold til contains (Sub tid er likt som de algoritmene den kaller)
	initArrayListQuick	O(n) i henhold til loadFile input
	qSortRunner	O(1)
	initArrBucket	O(n) i henhold til loadFile input
	bSortRunner	O(1)
	toIntArray	O(n)
	toArrayList	O(n)
BucketSortAlg	bucketSort	Værste :O(n2) Gjennomsnitt: O(n + k) Beste: O(n)
	testBucket	O(1) (sub tid er tiden bucketSort bruker)
QuickSortAlg	quickSort	Værste: O(n2) Gjennomsnitt: O(n log n) Beste: O(n log n)
	testQuickSort	O(1) (sub tid er tiden quicksort bruker)
	Get	O(1)
Edge	getId	O(1)
	Opposite	O(1) er bare sammenlikning
	endVertices	O(1)
	getElement	O(1)
	toString	O(n)
Node	addEdge	O(1)
	getParent	O(1)
	setParent	O(1)
	incidentEdges	O(1)
	Neighbors	O(1)
	isNeighbors	O(n)
	getId	O(1)
	getElement	O(1)
	toString	O(n)
Graph	addNode(id ,name)	O(1)
	addNode(v)	O(1)
	hasNodeId	0(1) p.g.a når addNode(v) blir brukt så blir noden lagret på index posisjonen, dette gjør at den kan hentes tilbake i konstant tid

	getNode	O(1) samme prinsippet her som hasNodeId, ved å lagre på index og hente på index
	addEdge	O(1)
	numVertices	O(1)
	containsNode	0(1) lagret på index, hente på index
	numOfEdges	O(1)
	Vetrices	O(1)