Tests de performances
Structures et algorithmes
Initiation à la Recherche

Marguerite Alain
Rincé Romain

Université de Nantes
2 rue de la Houssinière, BP92208, F-44322 Nantes cedex 03,
FRANCE

Table des matières

1 Chargement de pavages

Pour la génération des boîtes le nombre de caractéristiques est fixé à 20 et la dimension du pavage à 100. Ces valeurs ont été choisies car elles peuvent être considérées comme les valeurs au pire renvoyées par *Realpaver*. Cependant les caractéristiques pouvant être de différents types et ne connaissant pas la probabilité d'apparition de chacun de ces types, nous avons préféré construire des caractéristiques de chaque type en proportion un tiers.

1 Map internes

1.1 Pavage avec une première proposition de structure

Pour les Maps l'organisation des structures de données est la suivante :

- Il n'y a aucune structure de données globale, tout est stocké au niveau de la boîte. On a donc dans chaque boîte l'identifiant et la valeur de chaque caractéristique ou intervalle.
- Chaque boîte contient quatre Maps contenant les différentes informations :
  - Une Map pour l'ensemble des intervalles coordonnées.
  - Une Map pour les caractéristiques de type String.
  - Une Map pour les caractéristiques de type Interval.
  - Une Map pour les caractéristiques de type Number.

Les résultats obtenus sont indiqués dans la table A.1 et A.2 D'après les résultats, les deux structures fournissent un temps de construction et une occupation mémoire similaires. Il est encore difficile de déterminer laquelle de la TreeMap ou de la HashMap est la plus pertinente, les différences apparaitront sans doûte plus nettement au moment des tests d'accès aux caractéristiques.

1.2 Pavage avec une seconde proposition de structure

Pour les map, une autre organisation des structures de données possible est la suivante :

- Il y a une HashMap globale contenant le type de chaque caractéristique.
- Chaque boîte contient deux Maps contenant les différentes informations :
  - Une map pour l'ensemble des intervalles coordonnées.
  - Une liste pour l'ensemble des caractéristiques stockées dans la boîte sous la forme d'Objects.

Les résultats obtenus sont indiqués dans la table A.3 et A.4 Nous n'observons pas de différence nette, par rapport à la première proposition, quant au temps de construction et de l'occupation mémoire. Cette structure nécessitant d'effectuer un cast à chaque accès, il est plus avantageux de conserver la première structure.

2 Listes et Tableaux internes

Pour les structures liste et tableau l'organisation des structures de données est la suivante :

- Il y a une HashMap globale contenant le type de chaque caractéristique.
- Chaque boîte contient deux listes (ou tableaux) :
  - Une liste pour l'ensemble des intervalles coordonnées.
  - Une liste pour l'ensemble des caractéristiques de la boîte stockées sous la forme d'Objects

Les résultats obtenus sont indiqués dans la table A.5 et A.6 On peut constater que les structures de données en liste sont bien meilleures vis-à-vis du temps de construction et de l'espace mémoire(Tout du moins pour l'ArrayList) que les structures map de la première partie. Étrangement la structure LinkedList demande un espace mémoire presque 60% plus grand que celui alloué pour l'ArrayList.

Bien que l'ArrayList soit plus légère que les Maps de la prémière partie et plus rapide à construire, il est probable que celle-ci ne soit pas efficace pour l'accès. Nous traiterons cela dans une seconde étude.

3 Maps globales

Les boîtes sont composées de 4 ArrayLists :

- Une ArrayList pour l'ensemble des intervalles coordonnées.
- Une ArrayList pour les caractéristiques de type String.
- Une ArrayList pour les caractéristiques de type Interval.
- Une ArrayList pour les caractéristiques de type Number.

Il faut cependant savoir à quels indices se trouves les différents élément pour effectuer des accès directs. Pour cela, on dispose ici de quatre Maps globales :

- Une Map pour l'ensemble des intervalles coordonnées.
- Une Map pour les caractéristiques de type String
- Une Map pour les caractéristiques de type Interval
- Une Map pour les caractéristiques de type Number

La composition de chacune de ces maps est la suivante :

Clef : ID de la coordonée ou de la variable.

Valeur : Indice de l'objet dans le tableau de la boîte.

Observations : On propose trois cas de tests ou les Maps globales seront :

- Des HashMaps : A.7
- Des TreeMaps : A.8
- Une TreeMap construite à partir des HashMaps correspondantes : A.9

Les trois cas de tests nous renvoient des résultats relativement similaires mais qui se montrent aussi performants que les résultats de l'ArrayList. Il s'agit donc d'une organisation intéressante à étudier au niveau des temps d'accès.

2 Accès au élément d'une boîte

Dans cette partie nous allons effectuer des tests d'accès au éléments d'une boîte. Le nombre de coordonnées et de caractéristiques sont les même que dans la partie 1. De même le remplissage de chaque type de caractéristique demeure de un tiers pour chaque. Nous effectuerons d'abord des tests d'accès au caractéristiques puis des tests d'accès au coordonnées.

Il est aussi important de préciser que, pour tester, nous avons créer dix-mille boîtes et que pour accèder à une caractéristique, nous accèdons d'abord à une boîte aléatoirement. Les boîtes sont stockés dans un tableau à accès direct par indice. Même s'il est vrai que cette méthode introduit une constante dans le calcul de la complexité, elle permet de ne pas toujours accèder à la même information et donc de ne pas permettre un accès plus rapide qui ne serait pas réaliste.

1 Tests d'accès au caractéristiques

Dans tous les cas nous demanderons une caractéristiques en générant aléatoirement un identifiant.

1.1 Maps internes

Résultats tableau A.10

A Annexes

1 Études de performance avec des Maps

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.05s	52Mo	1Mo
1000	0.09s	52Mo	7Mo
10000	0.16s	119Mo	70Mo
100000	0.86s	776Mo	667Mo
1000000	Out of memory		

Table A.1 – Étude de performances avec des HashMap

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.06s	52Mo	1Mo
1000	0.07s	52Mo	6Mo
10000	0.16s	138Mo	64Mo
100000	0.84s	776Mo	635Mo
1000000	Out of memory		

Table A.2 – Étude de performances avec des TreeMap

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.06s	52Mo	1Mo
1000	0.08s	52Mo	7Mo
10000	0.13s	118Mo	69Mo
100000	0.84s	776Mo	655Mo
1000000	Out of memory		

Table A.3 – Étude de performances avec des HashMap seconde structure

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.07s	52Mo	1Mo
1000	0.08s	52Mo	6Mo
10000	0.15s	138Mo	64Mo
100000	0.88s	776Mo	633Mo
1000000	Out of memory		

TABLE A.4 – Étude de performances avec des TreeMap seconde structure

2 Études de performance avec des Tableaux et des Listes

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.05s	52Mo	<1Mo
1000	0.06s	52Mo	4Mo
10000	0.09s	66Mo	37Mo
100000	0.30s	408Mo	342Mo
1000000	Out of memory		

TABLE A.5 – Étude de performances avec des ArrayList

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.05s	52Mo	1Mo
1000	0.08s	52Mo	5Mo
10000	0.1s	88Mo	54Mo
100000	0.30s	776Mo	542Mo
1000000	Out of memory		

TABLE A.6 – Étude de performances avec des LinkedList

3 Études de performance avec des maps globales

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.05s	52Mo	0Mo
1000	0.07s	52Mo	4Mo
10000	0.1s	66Mo	38Mo
100000	0.34s	424Mo	351Mo
1000000	Out of memory		

Table A.7 – Étude de performances avec des HashMap globales

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.05s	52Mo	0Mo
1000	0.06s	52Mo	4Mo
10000	0.11s	66Mo	38Mo
100000	0.38s	422Mo	350Mo
1000000	Out of memory		

Table A.8 – Étude de performances avec des TreeMaps globales

Nombre de boîtes	CPU	Mémoire totale	Mémoire utilisée
100	0.05s	52Mo	0Mo
1000	0.06s	52Mo	4Mo
10000	0.11s	66Mo	38Mo
100000	0.39s	420Mo	350Mo
1000000	Out of memory		

Table A.9 – Étude de performances avec des TreeMaps construites à partir de HashMaps globales

Structure \ Nombre d'accès	10^6	10^7	10^8
HashMap	0.56s	5.24s	52.34s
TreeMap	0.53s	5.04s	48.75s

TABLE A.10 – Relevé des temps CPU d'accès au caractéristiques en secondes pour la HashMap et la TreeMap

Structure \ Nombre d'accès	10^6	10^7	10^8
HashMap	0.57s	5.34s	52.79s
TreeMap	0.58s	5.62s	54.81s

TABLE A.11 – Relevé des temps CPU d'accès au caractéristiques en secondes pour la HashMap et la TreeMap v2

Structure \ Nombre d'accès	10^6	10^7	10^8
ArrayList	0.43s	3.94s	39.15s
LinkedList	0.55s	5.17s	51.43s

TABLE A.12 – Relevé des temps CPU d'accès au caractéristiques en secondes pour l'ArrayList et la LinkedList