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 Génération de boîtes

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

1.1 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
- 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 Une seconde proposition de structure

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

- Il y a un TreeMap 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 Une fois encore, nous n'observons pas de différence nette quant au temps de construction et de l'occupation mémoire. Cette structure nécessitant d'effectuer un test à chaque accès, il est du coup préférable de conserver la première structure.

2 Liste et Tableau

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

- Il y a un TreeMap 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

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 et A.8
- Des TreeMaps : A.9 et A.10
- Une TreeMap construite à partir des HashMaps correspondantes : A.11 et A.12

Pour chaque cas, nous avons recommencé la série de tests en donnant aux constructeurs le nombre d'éléments qu'ils vont contenir. Lorsque cette information est fournie au constructeurs, les résultats révèlent dans les trois cas que la quantité de mémoire allouée est plus importante, mais celle réellement utilisée est plus faible. Par ailleurs on ne constate aucune différence flagrantes entre les trois structures. Pour 100000 boîtes elles nécessitent toutes les trois entre 310Mo et 350Mo. Les différences se manifesteront en théorie en cas d'accès à des boîtes.

A Annexes

1 Études de performance avec des Maps

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.02687s	0.05s	52Mo	1Mo
1000	0.05415s	0.09s	52Mo	7Mo
10000	0.36794s	0.16s	119Mo	70Mo
100000	10.50770s	0.86s	776Mo	667Mo
1000000	Out of memory			

Table A.1 – Étude de performances avec des HashMap

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.02873s	0.06s	52Mo	1Mo
1000	0.05472s	0.07s	52Mo	6Mo
10000	0.48741s	0.16s	138Mo	64Mo
100000	7.47501s	0.84s	776Mo	635Mo
1000000	Out of memory			

Table A.2 – Étude de performances avec des TreeMap

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.02201s	0.06s	52Mo	1Mo
1000	0.04329s	0.08s	52Mo	7Mo
10000	0.35066s	0.13s	118Mo	69Mo
100000	9.99475s	0.84s	776Mo	655Mo
1000000	Out of memory			

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

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.02792s	0.07s	52Mo	1Mo
1000	0.05299s	0.08s	52Mo	6Mo
10000	0.50868s	0.15s	138Mo	64Mo
100000	7.54041s	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	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.018991s	0.05s	52Mo	<1Mo
1000	0.03237s	0.06s	52Mo	4Mo
10000	0.11064s	0.09s	66Mo	37Mo
100000	1.85945s	0.30s	408Mo	342Mo
1000000	Out of memory			

Table A.5 – Étude de performances avec des ArrayList

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.02318s	0.05s	52Mo	1Mo
1000	0.05196s	0.08s	52Mo	5Mo
10000	0.21947s	0.1s	88Mo	54Mo
100000	6.12441s	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	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.00646	0.05s	52Mo	0Mo
1000	0.04284	0.07s	52Mo	4Mo
10000	0.146819	0.1s	66Mo	38Mo
100000	2.071468	0.34s	424Mo	351Mo
1000000	Out of memory			

TABLE A.7 – Étude de performances avec des HashMap globales sans initialisation

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.00237	0.05s	52Mo	0Mo
1000	0.03380	0.07s	52Mo	3Mo
10000	0.19556	0.09s	89mo	32mo
100000	2.0883	0.25s	480mo	312mo
1000000	Out of memory			

TABLE A.8 – Étude de performances avec des HashMap globales avec initialisation

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	.0074534	0.05s	52Mo	0Mo
1000	0.05651	0.06s	52Mo	4Mo
10000	0.14486s	0.11s	66Mo	38Mo
100000	2.10083s	0.38s	422Mo	350Mo
1000000	Out of memory			

TABLE A.9 – Étude de performances avec des TreeMaps globales sans initialisation

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.008355	0.05s	52Mo	7Mo
1000	0.03096	0.06s	52Mo	6Mo
10000	0.181671s	0.13s	89Mo	32Mo
100000	2.130028	0.30s	481Mo	318Mo
1000000	Out of memory			

TABLE A.10 – Étude de performances avec des TreeMaps globales avec initialisation

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.0085	0.05s	52Mo	0Mo
1000	0.05201	0.06s	52Mo	4Mo
10000	0.126349s	0.11s	66Mo	38Mo
100000	2.07826	0.39s	420Mo	350Mo
1000000	Out of memory			

TABLE A.11 – Étude de performances avec des TreeMaps construites à partir de HashMaps globales sans initialisation

Nombre de boîtes	Temps écoulé	CPU	Mémoire totale	Mémoire utilisée
100	0.008646	0.05s	52Mo	0Mo
1000	0.0.04127	0.07s	52Mo	6Mo
10000	0.18813s	0.10s	89Mo	32Mo
100000	2.12998599	0.29s	482Mo	317Mo
1000000	Out of memory			

TABLE A.12 – Étude de performances avec des TreeMaps construites à partir de HashMaps globales avec initialisation