# 宿題1.コンテナの種類と特徴

## 基本的な情報

* bitset  
  固定長配列，bit値を格納
* array  
  固定長配列，様々な型の値を格納
* vector  
  可変長配列，高速，メモリ効率が良い
* deque  
  可変長配列，末尾・先頭への挿入・削除が高速
* list  
  前後の要素同士を結合した双方向連結リスト
* forward\_list  
  前方のみと結合した片方向連結リスト
* set  
  要素の大小関係を自動で並び替え，分木に格納。大小のある型に対応，要素自体がキー
* multiset  
  多重集合（setと異なり，重複するデータの保持が可能）
* unodered\_set, unodered\_multiset  
  要素をキーのハッシュ値に基づくハッシュテーブルに格納，順序なし  
  (unordered\_set : 重複なし，unordered\_multiset : 重複あり)
* map  
  連想配列，要素を検索可能なキー(様々な型が可能)と紐付け
* multimap  
  連想配列，ひとつのキーに複数の対応する要素をもつことが可能
* unodered\_map, unodered\_multimap  
  連想配列，キーと要素のペアをハッシュで管理，順序なし

## ある要素への番号（またはキー）を指定したアクセスにかかる計算量

|  |  |
| --- | --- |
| 型 | 計算コスト(は格納された要素数) |
| vector |  |
| deque |  |
| unoredered\_map, unorederd\_multimap |  |
| map, multimap |  |
| list, forward\_list |  |

## 要素の挿入・削除にかかる計算量

|  |  |
| --- | --- |
| 型 | 計算コスト(は格納された要素数) |
| vector |  |
| deque |  |
| list, forward\_list |  |
| set, multiset |  |
| unoredered\_set, unorederd\_multiset |  |

：

末尾への削除，また末尾への挿入（平均的な場合）…

末尾への挿入（もっともコストの高い場合）…

上記以外への挿入・削除…

：

先頭・末尾への削除，また先頭・末尾への挿入（平均的な場合）…

先頭・末尾への挿入（もっともコストの高い場合）…

上記以外への挿入・削除…

## 要素の検索にかかる計算量

|  |  |
| --- | --- |
| 型 | 計算コスト(は格納された要素数) |
| unoredered\_set, unorederd\_multiset |  |
| set, multiset |  |
| vector |  |
| deque |  |
| list, forward\_list |  |
| map,multimap |  |
| unordered\_map, unordered\_multimap |  |

・ランダムアクセスと指定アクセスの違いは？

・使えるコマンドとコンテナの使い分けは？