자료구조 실습 보고서

[제 14-1 주] 사전의 성능 측정

제출일	2017/06/19
학 번	201000287
소 속	일어일문학과
이 름	유다훈

201000287 일어일문학과 유다훈

1 프로그램 설명서

1 주요 알고리즘 및 자료구조

- 알고리즘
  - 입출력
    - ◆ 입력은 없음.
      - i 필요한 데이터는 프로그램에서 생성
    - ◆ 출력
      - i 데이터 크기 변화에 따른 성능 측정 결과
- 자료구조
  - 정렬된 Array List
  - 정렬된 Linked List
  - Binary Search Tree

1 함수 설명서

Class	AppController			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppController()	없음	없음	_appView 변수를 초기화 하는 생성자 메소드
	void run()	없음	없음	성능 측정 프로그램을 실행하는 메소드
	void showExperimentByListOrderType(ListOrderanOrder)	enum	없음	들어오는 enum 타입에 따라 오름차순, 내림차순, 무작위 데이터 순으로 SortedArray, SortedLinkedList, BinarySearchTree 를 생성한다.
	void showUnitExperimentResult(UnitExperimentResultaResult)	실험결과	없음	전달받은 실험결과를 1000 으로 나누어 보기 편하게 출력한다.
	void showExperimentByDictionaryAndListOrderType (Dictionary<Integer, String> aDictionary, ListOrderanOrder)	Dictionary, enum	없음	오름차순, 내림차순, 무작위 차순에 따라 전달받은 자료구조형태의 사전의 실제 실험결과를 뽑아내는 메소드

Class	AppView			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppView()	없음	없음	생성자 메소드 값을 입력받는 스캐너 생성
	void output(String aString)	문자열	없음	줄바꿈을 하지 않는 메세지를 출력

	void outputLine(Stirng aString)	문자열	없음	줄바꿈을 하는 메세지를 출력.

Class	DataGenerator			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	static int[] ascendingOrderList(int aSize)	크기	정수형 배열	전달받은 크기만큼의 오름차순 데이터를 생성
	static int[] descendingOrderList(int aSize)	크기	정수형 배열	전달받은 크기만큼의 내림차순 데이터를 생성
	static int[] randomOrderList(int aSize)	크기	정수형 배열	전달받은 크기만큼의 무작위순 데이터를 생성

Class	ParameterSet			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	ParameterSet(int givenNumverOfExperiments, int givenFirstDataSize, int givenSizeIncrement)	측정횟수, 첫 데이터 크기, 증가량	없음	실험의 측정 횟수와 첫 실험 데이터 크기, 각 횟수마다의 증가량을 초기화하는 생성자 메소드
	int numberOfExperiments()	없음	측정횟수	측정회수를 리턴하는 메소드
	void setNumberOfExperiments(int newNumberOfExperiments)	측정회수	없음	새로운 측정회수를 설정하는 메소드
	int firstDataSize()	없음	없음	첫 데이터 크기를 리턴하는 메소드
	void setFirstDataSize(int newFirstDataSize)	첫 데이터 크기	없음	첫 데이터를 새롭게 설정하는 메소드
	int sizeIncrement()	없음	없음	각 횟수마다의 데이터 증가량을 리턴하는 메소드
	void setSizeIncrement(int newSizeIncrement)	데이터 증가량	없음	데이터 증가량을 새롭게 설정하는 메소드
	int maxDataSize()	없음	데이터의 전체 크기	데이터의 전체 크기를 계산하여 리턴하는 메소드

Class	PerformanceMesurement			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	PerformanceMeasurement()	없음	없음	ParameterSet 을 디폴트값으로 생성하고 데이터를 만드는 생성자 메소드
	PerformanceMeasurement(int giveNumberOfExperiments, int givenFirstDataSize, int givenSizeIncrement)	측정횟수, 첫 데이터 크기, 증가량	없음	전달받은 값들로 ParameterSet 을 초기화하고, 데이터를 생성하는 생성자 메소드
	ParameterSet parameterSet()	없음	ParameterSet	생성한 실험의 메타값을 전달한다.
	void setParameterSet(ParameterSet newParameterSet)	ParameterSet	없음	실험의 메타값을 새롭게 설정한다.
	int[] ascendingList()	없음	정수형 배열	오름차순의 배열을 리턴한다.
	int[] descendingList()	없음	정수형 배열	내림차순의 배열을 리턴한다.

	int[] randomList()	없음	정수형 배열	무작위순의 배열을 리턴한다.
	void generateData()	없음	없음	실험에 사용할 3 가지 데이터들을 최대크기로 생성해놓는다.
	int[] experimentList(ListOrder anOrder)	enum	정수형 배열	enum 형태에 맞는 실험용 데이터들을 리턴한다.
	UnitExperimentResult unitExperiment(Dictionary<Integer, String> aDictionary, ListOrder anOrder, int dataSize)	Dictionary, enum, 데이터 사이즈	측정 데이터	단위 측정 실험을 실행하고 그 결과를 리턴한다.
	UnitExperimentResult[] experimentByDictionaryAndListOrderType (Dictionary<Integer, String> aDictionary, ListOrder anOrder)	Dictionary, enum	측정 데이터들의 배열	데이터 크기별로 다시 단위 측정 실험을 실행하고 리턴한다

Class	UnitExperimentResult			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	UnitExperimentResult(int givenExperimentSize, double givenTimeForAdd, double givenTimeForSearch, double givenTimeForRemove	데이터 크기, 추가시간, 검색시간, 삭제시간	없음	전달받은 값으로 실험결과를 생성하는 생성자 메소드
	int experimentSize()	없음	데이터크기	데이터 크기를 리턴하는 메소드
	void setExperimentSize(int newExperimentSize)	데이터 크기	없음	전달받은 값으로 데이터 크기를 설정하는 메소드
	double timeForAdd()	없음	추가시간	데이터를 자료구조에 추가한 시간을 리턴하는 메소드
	void setTimeForAdd(double newTimeForAdd)	추가시간	없음	자료구조에 추가한 시간을 새롭게 설정하는 메소드
	double timeForSearch()	없음	검색시간	데이터를 자료구조에서 검색한 시간을 리턴하는 메소드
	void setTimeForSearch(double newTimeForSearch)	검색시간	없음	데이터를 자료구조에서 검색한 시간을 새롭게 설정하는 메소드
	double timeForRemove()	없음	삭제시간	데이터를 자료구조에서 삭제한 시간을 리턴하는 메소드
	void setTimeForRemove(double newTimeForRemove)	삭제시간	없음	데이터를 자료구조에서 삭제한 시간을 새롭게 설정하는 메소드

Class	LinkedNode			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	LinkedNode(T givenElement)	원소	없음	전달받은 원소로 노드를

				초기화하는 생성자 메소드
	LinkedNode(T givenElement, LinkedNode<T> givenNext)	원소, LinkedNode	없음	전달받은 원소와 LinkedNode 값으로 노드의 원소와 노드의 다음값을 생성하는 메소드
	T element()	없음	원소	원소를 리턴하는 메소드
	void setElement(T newElement)	원소	없음	전달받은 원소를 새롭게 설정하는 메소드
	LinkedNode<T> next()	없음	LinkedNode	현재 노드의 다음 노드값을 리턴하는 메소드
	void setNext(LinkedNode<T> newNext)	LinkedNode	없음	전달받은 노드값을 현재 노드의 다음 값으로 설정하는 메소드

Class	BinaryNode			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	BinaryNode()	없음	없음	BinaryNode 를 모두 null 값으로 초기화시키는 생성자 메소드
	BinaryNode(T givenElement, BinaryNode<T> givenLeft, BinaryNode<T> givneRight)	원소, 왼쪽노드, 오른쪽 노드	없음	전달받은 값으로 BinaryNdoe 의 원소와 왼쪽, 오른쪽 자식을 초기화하는 메소드
	T element()	없음	원소	원소를 리턴하는 메소드
	void setElement(T newElement)	원소	없음	현재 노드의 원소를 새롭게 지정하는 메소드
	BinaryNode<T> left()	없음	BinaryNode	현재노드의 왼쪽 노드를 리턴하는 메소드
	void setLeft(BinaryNode<T> newLeft)	BinaryNode	없음	현재 노드의 왼쪽 노드를 새롭게 지정하는 메소드
	BinaryNode<T> right()	없음	BinaryNode	현재 노드의 오른쪽 노드를 리턴하는 메소드
	void setRight(BinaryNode<T> newRight)	BinaryNode	없음	현재 노드의 오른쪽 노드를 새롭게 지정하는 메소드

Class	abstact Dictinary<K, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	int size()	없음	크기	사전의 크기값을 리턴하는 메소드
	void setSize(int newSize)	크기	없음	사전의 크기 값을 새롭게 설정하는 메소드
	Dictionary()	없음	없음	사전을 생성하는 생성자 메소드
	boolean isEmpty()	없음	boolean	사전이 비어있는지 없는지를 리턴하는 메소드
	abstract boolean isFull()	없음	boolean	사전이 꽉 찼는지 아닌지를 리턴하는 메소드
	abstract boolean keyDoesExist(K aKey)	Key	boolean	사전에 해당 키가 존재하는지 여부를 리턴하는 메소드

	abstract O objectForKey(K aKey)	key	Object	전달받은 키 값의 오브젝트를 리턴하는 메소드
	abstract boolean addKeyAndObject(K aKey, O anObject)	key, object	boolean	키와 오브젝트를 사전에 추가하고 추가 여부를 리턴하는 메소드
	abstract O removeObjectForKey(K aKey)	key	object	입력받은 키값과 해당 키값의 오브젝트를 삭제하는 메소드
	abstract void clear()	없음	없음	사전을 초기화시키는 메소드

Class	DictinaryElement <K extends Comparable<K>, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	DictionaryElement(K givenKey, O givenObject)	key, Object	없음	전달받은 키와 오브젝트로 사전의 원소를 생성하는 메소드
	K key()	없음	key	현재 원소의 키를 리턴하는 메소드
	void setKey(K newKey)	key	없음	현재 원소의 키를 새롭게 설정하는 메소드
	O object()	없음	Object	현재 원소의 오브젝트를 리턴하는 메소드
	void setObject(O newObject)	Object	없음	현재 원소의 오브젝트를 설정하는 메소드

Class	DictionaryBySortedArray<K extends Comparable<K>, O> extends Dictionary<K, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	DictionaryBySortedArray()	없음	없음	디폴트값으로 사전의 최대 크기와 배열의 크기를 생성하는 생성자 메소드
	DictionaryBySortedArray(int givenCapacity)	정수	없음	전달받은 값으로 사전의 최대 크기와 배열의 크기를 생성하는 생성자 메소드
	int capacity()	없음	정수	사전의 최대 용량을 전달하는 메소드
	void setCapacity(int newCapacity)	정수	없음	사전의 최대 용량을 새롭게 설정하는 메소드
	int positionFor(K aKey)	key	정수	전달받은 키를 검색하여 배열 내에 키가 어디에 위치하고 있는지 리턴하는 메소드
	void makeRoomAt(int aPosition)	정수	없음	해당 정수값의 위치까지의 배열의 데이터들을 한 칸씩 뒤로 이동시키는 메소드
	void removeGapAt(int aPosition)	정수	없음	해당 정수값의 위치까지 배열의 데이터들을 한칸씩 앞으로 이동시키는 메소드

Class	DictionaryBySortedLinkedList<K extends Comparable<K>, O> enxtends Dictionary<K, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	DictionaryBySortedLinkedList()	없음	없음	LinkedList 를 초기화시키는 메소드
	LinkedNode<DictionaryElement<K, O>> head()	없음	LinkedNode	LinkedList 의 헤드값을 리턴하는 메소드
	void setHead(LinkedNode<DictionaryElement<K, O>> newHead)	LinkedNode	없음	LinkedList 의 헤드값을 새롭게 설정하는 메소드

Class	DictionaryByBinarySearchTree<K extends Comparable<K>, O> extends Dictionary<K, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	DictionaryByBinarySearchTree()	없음	없음	BinarySearchTree 를 초기화시키는 메소드
	BinaryNode<DictinaryElement<K,O>> root()	없음	BinaryNode	트리의 루트값을 리턴하는 메소드
	void setRoot(BinaryNode<DictinaryElement<K,O>> newRoot)	BinaryNode	없음	트리의 루트값을 새롭게 설정하는 메소드
	DictionaryElement elementForKey(K aKey)	Key	DictionaryElement	키가 들어있는 노드를 찾아 해당 키와 맞는 노드의 원소를 리턴한다.
	DictionaryElement<K, O> removeRightMostElementOfLeftSubTree (BinaryNode<DictinaryElement<K,O>> root)	root	DictionaryElement	왼쪽 서브트리의 최대값을 찾아 삭제하는 메소드

2 종합 설명서

- 프로그램을 실행하면 오름차순, 내림차순, 무작위순의 데이터가 추가적인 입력없이 자동으로 생성된다.
- 해당 데이터들을 세 개의 자료구조에 넣으면서 오름차순으로 정렬하며, 원하는 키를 검색하고, 다시 삭제하는 동안의 각각의 자료구조에서 걸리는 시간을 측정한다.
  - PerformanceMeasurement 메소드에서 측정.
  - 정렬된 배열, 정렬된 연결체인, 이진탐색트리.

2 프로그램 장단점 분석

- 장점
  - 오름차순, 내림차순, 무작위 순의 데이터를 세 가지의 자료구조에 다시 오름차순으로 정렬해가며 삽입하는 시간, 특정 값을 검색하는 시간, 삭제하는 시간을 비교하여 어떠한 상황에 어떠한 자료구조 제일 적합한지를 파악할 수 있다.
- 단점
  - 프로그램을 구현하면서, 이해하기가 어렵다.

3 실행 결과 분석

1 입력과 출력

프로그램 실행

```
<<"Dictionary" 의 성능 측정을 시작합니다.>>
```

출력 1-1 - 오름차순 데이터를 사용한 측정 : SortedArrayList

```
<오름차순 데이터를 사용한 측정 (단위: micro second) >
"DictionaryBySortedArray"로 구현된 사전의 성능
    크기              삽입              검색              삭제
[10000]             811              1013             36237
[20000]            2014              2337            147983
[30000]            3296              3239            448040
[40000]            3639              4113            780048
[50000]            5205              6021           1191644
```

출력 1-2 - 오름차순 데이터를 사용한 측정 : SortedLinkedList

```
"DictionaryBySortedLinkedList"로 구현된 사전의 성능
    크기              삽입              검색              삭제
[10000]           243773            234478               890
[20000]          1013266             996137              1047
[30000]          2147483            2147483              1272
[40000]          2147483            2147483              1886
[50000]          2147483            2147483              1970
```

출력 1-3 - 오름차순 데이터를 사용한 측정 : BinarySearchTree

```
"DictionaryByBinarySearchTree"로 구현된 사전의 성능
    크기              삽입              검색              삭제
[10000]           251110            268405              2095
[20000]          1082232           1075457              1336
[30000]          2147483           2147483              1462
[40000]          2147483           2147483              2143
[50000]          2147483           2147483              2198
```

출력 2-1 – 내림차순 데이터를 사용한 측정 : SortedArrayList

```
<내림차순 데이터를 사용한 측정 (단위: micro second) >
"DictionaryBySortedArray"로 구현된 사전의 성능
    크기              삽입              검색              삭제
[10000]          45113          1170           772
[20000]         198835          1951          1499
[30000]         434002          3325          2542
[40000]         780387          5610          3340
[50000]        1196760          5285          4333
```

출력 2-2 - 내림차순 데이터를 사용한 측정 : SortedLinkedList

```
"DictionaryBySortedLinkedList"로 구현된 사전의 성능
    크기              삽입              검색              삭제
[10000]            479        228921        238577
[20000]            936        718063        711465
[30000]           1422       2147483       2147483
[40000]           1977       2147483       2147483
[50000]           2839       2147483       2147483
```

출력 2-3 - 내림차순 데이터를 사용한 측정 : BinarySearchTree

```
"DictionaryByBinarySearchTree"로 구현된 사전의 성능
    크기              삽입              검색              삭제
[10000]         278911         280163           610
[20000]        1023670         996275           870
[30000]        2147483        2147483          1244
[40000]        2147483        2147483          2337
[50000]        2147483        2147483          2369
```

출력 3-1 - 무작위 데이터를 사용한 측정 : SortedArrayList

```
<무작위 데이터를 사용한 측정 (단위: micro second) >
"DictionaryBySortedArray"로 구현된 사전의 성능
    크기            삽입            검색            삭제
[10000]         25943          1830          22622
[20000]         93176          4720          81059
[30000]        217646          7903         172294
[40000]        375770         10121         313423
[50000]        590424         13815         500281
```

출력 3-2 - 무작위 데이터를 사용한 측정 : SortedLinkedList

```
"DictionaryBySortedLinkedList"로 구현된 사전의 성능
    크기            삽입            검색            삭제
[10000]        233016         592134         219406
[20000]       1220371        2147483        1295672
[30000]       2147483        2147483        2147483
[40000]       2147483        2147483        2147483
[50000]       2147483        2147483        2147483
```

출력 3-3 - 무작위 데이터를 사용한 측정 : BinarySearchTree

```
"DictionaryByBinarySearchTree"로 구현된 사전의 성능
    크기            삽입            검색            삭제
[10000]          2033           2368           1906
[20000]          5425           6056           5112
[30000]         10246           8247           9419
[40000]         15123          10801          14328
[50000]         16284          18250          16479
```

프로그램 종료

```
<<"Dictionary" 의 성능 측정을 종료합니다. >>
```

2 결과 분석

- 오름차순
  - Sorted Array List
    - i 오름차순인 데이터를 그냥 오름차순으로 삽입하기 때문에 삽입속도가 매우 짧다.
    - ii 이미 정렬 되어있는 데이터에서 특정 데이터를 검색하기 위해 검색을 하는 positionFor()메소드의 경우, 전체 사이즈에서 중간값을 정하여 큰 값은 배열의 오른쪽, 작은 값은 배열의 왼쪽으로 진행하는 단순한 원리로 인하여, 검색 속도가 짧다.
    - iii 삭제 시 원하는 데이터 삭제를 하고 해당 데이터 이후의 배열의 모든 데이터를 한 칸씩 이동을 시켜주어야 하기 때문에 시간이 걸린다.
  - Sorted Linked List
    - i 오름차순을 삽입 할 때에는 항상 현재 노드의 뒷부분에 삽입을 해줘야 하기 때문에, 값이 커지면 커질수록 시간이 오래걸린다.
    - ii 원하는 값을 검색하는 경우에도 모든 노드를 원하는 값이 나올 때까지 하나하나 확인을 해야하기 때문에 오래 걸린다.
    - iii 삭제를 할 때에는 오름차순의 데이터를 입력받았을 때, 이미 자료구조는 오름차순 순이므로, LinkedList 의 특성상 앞의 head 만 삭제하고 뒤 노드들을 앞으로 당겨주기만 하면 되므로 시간이 오래걸리지 않는다.
  - Binary Search Tree
    - i 이진탐색트리의 특성상 현재 노드값보다 큰 값은 무조건 오른쪽으로만 가게된다. 즉 오름차순 데이터를 입력받으면 무조건 오른쪽 서브트리만 생기게 된다. 이것은 연결체인의 구조와 똑같은 구조이다. 삽입시간이 연결체인만큼 걸리는 것도 같은 구조이기 때문이다.
    - ii 검색 역시 원하는 값을 찾을 때까지 모든 노드를 검색해야하는 연결체인과 같은 구조이다. 시간이 오래걸린다.
    - iii 삭제 시의 현재 삭제를 원하는 노드가 헤드인지, 혹은 노드에 자식이 오른쪽에만 있는지, 왼쪽에만 있는지, 양쪽 다 있는지 확인해야하기 때문에, 연결체인과 같은 구조이지만 삭제시 시간이 연결체인보다 약간 더 걸린다.
- 내림차순
  - Sorted Array List
    - i 삽입 시에 현재 들어있는 데이터를 한 칸씩 뒤로 미루면서 삽입해야 하기 때문에 시간이 오래걸린다.
    - ii 중간값을 정하여 큰 값은 배열의 오른쪽, 작은 값은 배열의 왼쪽으로

진행하는 알고리즘으로 인해 검색시간이 짧다.

iii 내림차순으로 들어오는 데이터를 삭제할 때에는, 자료구조 내에서 정렬된 상태에서 데이터를 삭제하는 것이다. 내림차순 데이터에서 제일 큰 값은 배열의 제일 뒤에 있으므로 나머지 배열의 데이터에 대해서 공간을 앞뒤로 밀거나 당길 필요가 없어서 삭제 시간이 빠르다.

- Sorted Linked List
  - i 값을 비교하여 삽입할 때, 무조건 이전 값보다 작은값이 들어오기 때문에, head 의 위치에만 넣어주면 되므로 시간이 짧게 걸린다.
  - ii 모든 노드를 검색해야하기 때문에 검색 시간이 오래걸린다. 더군다나 삽입되는 데이터는 연결 체인의 뒷부분에 있으므로 무조건 모든 노드를 검색 해야하며 그만큼 시간이 오래 걸린다.
  - iii 삭제도 검색과 같은 이유로 인하여 시간이 오래 걸린다.
- Binary Search Tree
  - i 값을 비교하여 삽입할 때, root 보다 작은 값은 무조건 왼쪽의 자식이 되므로, 연결체인과 같은 한 방향 트리가 생성된다. 자식으로 지정하는 것을 모든 노드를 거쳐가며 노드의 맨 끝에서 왼쪽자식으로 지정해주는 것이므로 시간이 오래 걸린다.
  - ii 검색 역시 원하는 값을 찾기 위해 모든 노드를 방문해야한다.
  - iii 삭제 시의 이진탐색트리는 내림차순으로 정렬되어있기 때문에, 내림차순의 데이터가 들어오면 루트 노드만 반환해주면 되므로 시간이 빠르다.

● 무작위순

- Sorted Array List
  - i 무작위로 삽입되는 데이터들을 배열내에 삽입되었던 데이터와 비교하고, 그냥 넣을지 아니면 공간을 만들어 데이터와 데이터 사이에 넣을지의 문제가 발생하므로 삽입 시 시간이 걸린다. 그러나 모든 데이터를 계속해서 뒤로 밀어야 하는 내림차순보다는 시간이 덜 걸린다.
  - ii 데이터를 검색 한 후 찾은 인덱스로 바로 검색을 할 수 있다.
  - iii 무작위로 들어온 데이터를 찾아서 삭제 후, 해당 데이터의 뒤의 공간에 있는 데이터를 앞으로 당길지 말지의 여부때문에 삭제 또한 삽입과 비슷한 시간이 걸린다.
- Sorted Linked List
  - i 무작위로 들어온 데이터를 삽입하기 위해서 모든 노드들과 하나하나 비교를 해야하기 때문에, 데이터가 많아지면 많아질 수록 시간이 오래걸린다.
  - ii 검색 역시 원하는 데이터를 찾을 때 까지 연결체인을 검색해야만

한다.

iii 삭제 또한 원하는 데이터를 찾을때까지 연결체인을 검색해야만 한다.

- Binary Search Tree

i 삽입 시에 루트노드와 비교하여 루트노드보다 작으면 왼쪽, 크면 오른쪽으로 보내고, 해당 자리가 null 이면 그 자리에 지정을 하는 단순한 원리때문에 시간이 적게 걸린다.

ii 검색 또한 루트노드와만 비교하면 되므로 시간이 덜 걸린다.

iii 삭제 또한 루트노드와 비교하고 서브트리에서 값을 끌어 올릴지 말지의 여부만 정하면 되므로 시간이 적게 걸린다.

- 결론

- Sorted Array List

	오름차순	내림차순	무작위
삽입	O(1)	O(n)	O(n)
검색	O(1)	O(1)	O(1)
삭제	O(n)	O(1)	O(n)

- Sorted Linked List

	오름차순	내림차순	무작위
삽입	O(n)	O(1)	O(n)
검색	O(n)	O(n)	O(n)
삭제	O(1)	O(n)	O(n)

- Binary Search Tree

	오름차순	내림차순	무작위
삽입	O(n)	O(n)	O(log n)
검색	O(n)	O(n)	O(log n)
삭제	O(1)	O(1)	O(log n)

- 오름차순 데이터를 이용한 성능 분석 시 Array List 가 좋다.(삽입과 검색)
- 내림차순을 이용한 성능 분석 시 Array List 가 좋다. (검색과 삭제)
- 무작위 데이터를 이용한 성능 분석 시 Binary Search Tree 가 모든 면에서 월등히 좋다.
- 클래스의 상속을 통해 size()메소드를 각각의 자료구조마다 구현하지 않고 사용할 수 있었으며, 또한 각각의 자료구조마다 똑같은 파라미터를 가진 메소드를 구현하여 손쉽게 구현할 수 있었다.
- Binary Search Tree 는 노드마다의 비교때문에 구현해야할 코드의 량이 길고 복잡했지만 성능이 월등히 우수하다.