Today

- Key-Value Pair Collections

Assignments

- HW 4 due
- HW 5 out
Key-Value Pair Collections (HW-5 and beyond)

Managing collections of key-value pairs
- A "key" represents an identifier
- Each key has a corresponding value
- Together they form a "key-value" pair

Examples of data modeled as key-value pairs:
- Stock symbol (key) to stock price (value)
- An SSN (key) to a name (value)
- A state (key) to its area, capital, total population ("record" value)
- URL (webpage) to its keywords (list), or vice versa

Key-Value Collection ADT
- Also called a Dictionary, Map (HW-5), Associative Array
- Many potential data structure implementations
- We’ll require unique key values per collection

Many commercial software products and uses, e.g.:
- In-memory content caching (e.g., for large web sites)
- Storing user session data
- Serving real-time recommendations / ads
- Big data distributed queries / computation (e.g., Hadoop)
Our Map ADT:

```cpp
template<typename K, typename V>
class Map
{
public:

  // get number of key-value pairs and check if empty
  virtual int size() const = 0;
  virtual bool empty() const = 0;

  // update and return a key's value
  virtual V& operator[](const K& key) = 0;
  virtual const V& operator[](const K& key) const = 0;

  // add and remove a key value pair (assumes unique keys)
  virtual void insert(const K& key, const V& value) = 0;
  virtual void erase(const K& key) = 0;

  // check if the map contains a pair with given key
  virtual bool contains(const K& key) const = 0;

  // keys k in the collection such that k1 <= k <= k2
  virtual ArraySeq<K> find_keys(const K& k1, const K& k2) const = 0;

  // all map keys in ascending (sorted) order
  virtual ArraySeq<K> sorted_keys() const = 0;
};
```

For example:

```cpp
Map<string, int>& m = ...; // must be a concrete type
m.insert("the", 27);
m.insert("and", 20);
if (m.contains("and")) // true
  // ...
cout << m["the"] << endl; // outputs 27
m["and"] = 21; // change "and" value
  // ...
ArraySeq<string> s1 = m.sorted_keys();
  // ...
ArraySeq<string> s2 = m.find_keys("a", "t"); // "and" but not "the"
```
Map Implementations in HW-5

For each implementation, we’ll use `std::pair<K,V>`

- has a `first` and `second` member variable
- to declare: `std::pair<K,V> p;`
- to declare and initialize: `std::pair<K, V> {key, value};`
- to access the “key”: `K key = p.first;`
- to access the “value”: `V val = p.second;`
- to update the “value”: `p.second = value;`
- note: for `operator[]`, we will: `return p.second;`

(1). ArrayMap:

- uses `ArrayList` to implement the `Map` functions
- a private member variable `ArraySeq<std::pair<K,V>> seq;`
- insert into the `Map` at the end of seq
- iterate through `seq` for: `operator[]`, `erase()`, `contains()`, `find_keys()`
- for `sorted_keys()`, create temporary `ArraySeq<K>` from `seq` keys, call `sort()` on temporary, then return temporary

(2). LinkedMap:

- works same as `ArrayMap` but uses `LinkedSeq<std::pair<K,V>> seq;`
- note: sorted key works the same
(3). BinSearchMap:

- Uses an underlying `ArraySeq<std::pair<K,V>> seq;` member variable

- Goal is to optimize `contains` by using binary search
  - To do this, we need to keep the underlying `seq` sorted
  - `sorted_keys()` also won’t need to call `ArraySeq::sort()`

- Must implement a private helper function:
  ```cpp
  bool bin_search(const K& key, int& index) const;
  ```

- The helper iteratively finds index of given key via binary search
  - if the key isn’t in the map, returns “closest” index
  - but, always returns a valid index (for a non-empty sequence)

- Must use `bin_search` helper to implement:
  - `contains` ... just return result of `bin_search`
  - `insert` ... find index to insert at (roughly)
  - `erase` ... find index to erase
  - `find_keys` ... find index for first key of range
  - `operator[]` ... find index of pair (to return pair’s value)

- Q: What are the expected performance improvements and trade-offs?
  - faster `contains`, `erase`, and `sorted_keys`
  - faster `find_keys` depending on “selectivity” of range
  - slower `insert` compared to just inserting at end of `seq`

We’ll go over binary search in a bit, with analysis