4.5 Symbol Table Applications

Set ADT

Set. Unordered collection of distinct keys.

API for SET.

- `add(key)` insert the key into the set
- `contains(key)` is the given key in the set?
- `remove(key)` remove the key from the set
- `iterator()` return iterator over all keys

Q. How to implement?

Java library. `java.util.HashSet`.

Set Client: Remove Duplicates

Remove duplicates. [e.g., from commercial mailing list]

- Read in a key.
- If key is not in set, insert and print it out.

```
public class DeDup {
   public static void main(String[] args) {
      SET<String> set = new SET<String>();
      while (!StdIn.isEmpty()) {
         String key = StdIn.readString();
         if (!set.contains(key)) {
            set.add(key);
            System.out.println(key);
         }
      }
   }
}
```


More Set Applications

Application	Purpose	Key
Spell checker	Identify misspelled words	Word
Browser	Highlight previously visited pages	URL
Chess	Detect repetition draw	Board position
Spam blacklist	Prevent spam	IP address
Security whitelist	Allow trusted traffic	IP address
Credit card fraud	Identify stolen credit cards	Credit card number

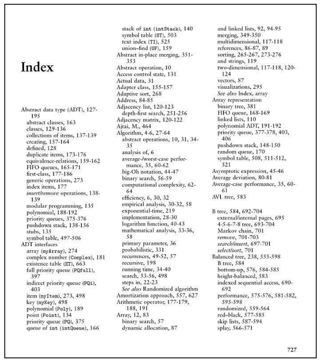
Inverted Index

Inverted Index

Inverted index. Given a list of page, preprocess so that you can quickly find all pages containing a given query.

Ex 1. Book index.
Ex 2. Web search engine index.
Ex 3. File index, e.g., Spotlight.

Key. Query word.
Value. Set of pages.



Inverted Index Implementation

```
public class InvertedIndex {
   public static void main(String[] args) {
      ST<String, SET<String>> st;            // use SET to avoid dups
      st = new ST<String, SET<String>>();

      for (String filename : args) {
         In in = new In(filename);
         while (!in.isEmpty()) {
            String word = in.readString();
            if (!st.contains(word))
               st.put(word, new SET<String>());
            SET<String> set = st.get(word);
            set.add(filename);
         }
      }                                       // build inverted index

      while (!Stdin.isEmpty()) {
         String query = StdIn.readString();
         System.out.println(st.get(query));
      }                                       // process queries
   }
}
```


Inverted Index

Extensions.

- Ignore stopwords: `the`, `on`, `of`, etc.
- Multi-word queries:
  - set intersection (AND).
  - set union (OR).
- Record position and number of occurrences of word in document.

Sparse Vectors and Matrices

Vectors and Matrices

Vector. Ordered sequence of N real numbers.
Matrix. N-by-N table of real numbers.

$$a = [\,0 \;\; 3 \;\; 15\,], \quad b = [-1 \;\; 2 \;\; 2\,]$$
$$a + b = [-1 \;\; 5 \;\; 17\,]$$
$$a \circ b = (0 \cdot -1) + (3 \cdot 2) + (15 \cdot 2) = 36$$

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad A + B = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 6 & -2 \\ 0 & 3 & 18 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix} \times \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$$

Sparsity

Def. An N-by-N matrix is sparse if it has O(N) nonzeros entries.

Empirical fact. Large matrices that arise in practice are usually sparse.

Matrix representations.
- 2D array: space proportional to N^2.
- Goal: space proportional to number of nonzeros, without sacrificing fast access to individual elements.

Ex. Google performs matrix-vector product with N = 4 billion!

Sparse Vector Implementation

```
public class SparseVector {
   private final int N;
   private ST<Integer, Double> st = new ST<Integer, Double>();

   public SparseVector(int N) { this.N = N; }

   public void put(int i, double value) {
      if (value == 0.0) st.remove(i);
      else              st.put(i, value);
   }

   public double get(int i) {
      if (st.contains(i)) return st.get(i);
      else                return 0.0;
   }
```

a[i] = value

a[i]

Sparse Vector Implementation (cont)

```
// return a · b
public double dot(SparseVector b) {
   SparseVector a = this;
   double sum = 0.0;
   for (int i : a.st)
      if (b.st.contains(i)) sum += a.get(i) * b.get(i);
   return sum;
}
```

```
// return c = a + b
public SparseVector plus(SparseVector b) {
   SparseVector a = this;
   SparseVector c = new SparseVector(N);
   for (int i : a.st) c.put(i, a.get(i));
   for (int i : b.st) c.put(i, b.get(i) + c.get(i));
   return c;
}
```


Sparse Matrix Implementation

```
public class SparseMatrix {
   private final int N;
   private SparseVector[] rows;

   public SparseMatrix(int N) {
      this.N  = N;
      rows = new SparseVector[N];
      for (int i = 0; i < N; i++)
         rows[i] = new SparseVector(N);
   }

   public void put(int i, int j, double value) {
      rows[i].put(j, value);
   }

   public double get(int i, int j) {
      return rows[i].get(j);
   }
```

N-by-N matrix of all zeros

A[i][j] = value

A[i][j]

Sparse Matrix Implementation (cont)

```
// return b = A*x
public SparseVector times(SparseVector x) {
   SparseMatrix A = this;
   SparseVector b = new SparseVector(N);
   for (int i = 0; i < N; i++)
      b.put(i, rows[i].dot(x));
   return b;
}
```

```
// return C = A + B
public SparseMatrix plus(SparseMatrix B) {
   SparseMatrix A = this;
   SparseMatrix C = new SparseMatrix(N);
   for (int i = 0; i < N; i++)
      C.rows[i] = A.rows[i].plus(B.rows[i]);
   return C;
}
```


Similarity Search

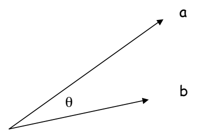
Similarity Measure

Dot product. Given two vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, their dot product $a \cdot b = a_1 b_1 + \ldots + a_n b_n$.

Magnitude. The magnitude of vector a is given by $|a| = \sqrt{a \cdot a}$.

Similarity. Given two vectors a and b, $\cos\theta = \frac{a \cdot b}{|a|\,|b|}$.

- Similar if $\cos\theta$ close to 1.
- Not similar if $\cos\theta$ close to 0.



Applications. Literature, code, music, video, artwork.

A Plan for Spam

A Plan for Spam

Bayesian spam filter.

- Filter based on analysis of previous messages.
- User trains the filter by classifying messages as spam or ham.
- Parse messages into tokens (alphanumeric, dashes, ', $).

Build data structures.

- Symbol table A of tokens and frequencies for spam.
- Symbol table B of tokens and frequencies for ham.
- Symbol table C of tokens with prob p that they appear in spam.

```
double h = 2.0 * ham.freq(word);
double s = 1.0 * spam.freq(word);
double p = (s/spams) / (h/hams + s/spams);
```

bias probabilities to avoid false positives

Reference: http://www.paulgraham.com/spam.html

A Plan for Spam

Identify incoming email as spam or ham.

- Find 15 most interesting tokens (difference from 0.5).
- Combine probabilities using Bayes law.

which data structure?

$$\frac{p_1 \times p_2 \times \cdots \times p_{15}}{(p_1 \times p_2 \times \cdots \times p_{15}) + ((1-p_1)\times(1-p_2)\times\cdots\times(1-p_{15}))}$$

- Declare as spam if threshold > 0.9.

Details.

- Words you've never seen.
- Words that appear in ham corpus but not spam corpus, vice versa.
- Words that appear less than 5 times in spam and ham corpuses.
- Update data structures.