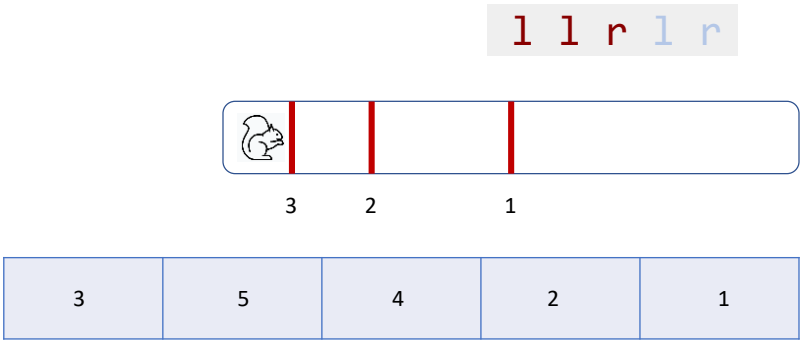
Herbert Wertheim
College of Engineering
Department of Computer & Information Science & Engineering
UNIVERSITY of FLORIDA

Data Structures – Part II

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

Example - Escape from Stones

- Question
  - https://codeforces.com/problemset/problem/264/A



Set/Treeset

- In C++, set is ordered and based on a balanced search tree and its operations work in O(log n) time
- unsorted_set: in C++, unsorted_set is based on a hash table and its operations work on O(1) on average
- C++ STL set, similar to C++ STL map
  - map stores a (key, data) pair
  - set stores just the key
- In Java: TreeSet based on a self-balancing tree

- Example: UVa10815 - Andy's First Dictionary

Pseudocode

foreach row of input
 convert all character to lower cases
  and change all non-alphabet character to space
 split into words and insert to set

print out word from set

```
set<string> wordList;
while (cin >> s) {
     for (int i = 0; i < s.size(); i++) {
         if (isalpha(s[i])) {
             s[i] = tolower(s[i]);
         }
         else
             s[i]=' ';
     }
     stringstream s2(s);
     while(s2 >> s1)
        wordList.insert(s1);
 }
 set<string>::iterator ptr;
 for (ptr = wordList.begin(); ptr != wordList.end(); ptr++) {

         cout << *ptr << endl;

 }
```

Maps

- Associative containers that store elements in a mapped fashion
  - Each element has a key value and a mapped value
  - No two mapped values can have same key values
- map is based on a balanced binary search tree and its operations work in O(log n) time
  - C++ STL map (Java TreeMap)
- Example
  - Given an array, find the k-th occurrence (from left to right) of an integer v

8 3
1 3 2 2 4 3 2 1
1 3 → 2
2 4 → 0
3 2 → 7

$1 \le n, m \le 100,000, 1 \le k \le n, 1 \le v \le 1,000,000$.


```
map<int, vector<int> > a;

for(int i = 0; i < n; i++) {
  cin >> x;
  if(!a.count(x))
    a[x] = vector<int>();
  a[x].push_back(i+1);
}

while(m--) {
  cin >> x >> y;
  if(!a.count(y) || a[y].size() < x)
    cout << "0" << endl;
  else
    cout <<  a[y][x-1] << endl;
}
```

Priority Queue

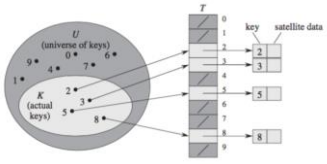
- A multiset designed such that the first element of the queue is the greatest of all elements in the queue and elements are in non increasing order
  - each element of the queue has a priority
    - fixed order
  - Smaller constant factor than multiset
  - Based on heap structure, which is a special binary tree
  - Used when you only needs to find minimum or maximum value
  - Descending order
    - largest value first
- UVA 1203 Argus
  - Given a number of tasks with id number and an interval, and a number k, print out the first k tasks to return in chronological order


```
priority_queue< pair< pair<int, int>, int> > pq;

while(true) {
   cin >> s;
   if(s == "#") break;
   cin >> queryNum >> period;
   pq.push(make_pair(make_pair(-period, -queryNum), -period));
}
cin >> k;
while(k--) {
   auto event = pq.top();
   pq.pop();
   cout << -event.first.second << '\n';
   event.first.first += event.second;
   pq.push(event);
}
```

Hash table

- Advertised O(1) for insert, search, and delete, but
  - The hash function must be good!
  - There is no Hash Table in C++ STL (Yes in Java
- Nevertheless, O(log n) using map is usually
- Direct Addressing Table (DAT)
  - Key values are distinct, and is drawn from a universe U = {0, 1, . . . , m - 1}
  - Store the items in an array, indexed by keys



- Example
  - UVa 11340 (Newspaper)

```
map <char, double> charMap;
charMap[c] = v;

value = charMap.find(ch);
if(value != charMap.end())
        total += value->second;
```

Faster way?

```
int map[522];
map[c+256] = v;
```


Comparisons – in C++

- Count unique elements

Table 5.1 Results of an experiment where the number of unique elements in a vector was calculated. The first two algorithms insert the elements to a set structure, while the last algorithm sorts the vector and inspects consecutive elements

Input size n	set (s)	unordered_set (s)	Sorting (s)
10^6	0.65	0.34	0.11
$2 \cdot 10^6$	1.50	0.76	0.18
$4 \cdot 10^6$	3.38	1.63	0.33
$8 \cdot 10^6$	7.57	3.45	0.68
$16 \cdot 10^6$	17.35	7.18	1.38

- Add/Remove elements

Table 5.3 Results of an experiment where elements were added and removed using a multiset and a priority queue

Input size n	multiset (s)	priority_queue (s)
10^6	1.17	0.19
$2 \cdot 10^6$	2.77	0.41
$4 \cdot 10^6$	6.10	1.05
$8 \cdot 10^6$	13.96	2.52
$16 \cdot 10^6$	30.93	5.95

- Determine the most frequent value

Table 5.2 Results of an experiment where the most frequent value in a vector was determined. The two first algorithms use map structures, and the last algorithm uses an ordinary array

Input size n	map (s)	unordered_map (s)	Array (s)
10^6	0.55	0.23	0.01
$2 \cdot 10^6$	1.14	0.39	0.02
$4 \cdot 10^6$	2.34	0.73	0.03
$8 \cdot 10^6$	4.68	1.46	0.06
$16 \cdot 10^6$	9.57	2.83	0.11

Prefix Sum Array

- The sums of prefixes (running totals) of the input sequence:

input wordListbers	1	2	3	4	5	6	...
prefix sums	1	3	6	10	15	21	...

$PreSum_0 = a_0$
$PreSum_1 = a_0 + a_1 = PreSum_0 + a_1$
$PreSum_2 = a_0 + a_1 + a_2 = PreSum_1 + a_2$
. . .
$PreSum_n = PreSum_{n-1} + a_n$

```
1  n = int(input())
2  prefix = []
3  s = 0
4  res = 0
5
6  for i in input().split():
7      s += int(i)
8      prefix.append(s)
9
10 for i in range(n - 1):
11     if s == 2*prefix[i]:
12         res += 1
13
14 print(res)
```

- Example:
  - Stripe: https://codeforces.com/problemset/problem/18/C