Programming Competition

Coaching sessions

Coach: Francisco R. Ortega

Session 1

Sessions

- Most sessions will be 2 hours
  - 2 to 4pm Mondays
- Do you guys want to do video conferencing during weekend to solve problems?
  - Oovoo or Skype
- We will have 30 minutes to 1 hour discussions
- Then we will try to solve some problems

Logistics

- We meet every Monday from 2 to 4pm
  - October 15 will not have a meeting.
- You can find me in Skype.
  - Ask for my username
- Email me if you have questions.
- Find a time where you can meet with your team.
- How long would it take to cover the whole book?
  - Maybe more than one semester

Practice Competition

- We will have our own competition
  - Hopefully teams of 2 or 1
- A few problems based on material covered
- Towards the end of this year (2012)
- Winner(s) will get a prize yet to be determined
  - For example: BIG USB Flash Drive.

Few Tips

- Submit problems to uva.onlinejudge.org
- Type Fast (if you don't type fast, practice)
- Know your Data Structures in STL
- Master your programming language
  - I will use C++ (sometimes Java)
- Purchase Competitive Programming 2 Book

Path

- We will begin with Chapter 2.
  - Read Chapter 1.
    - At some point we will go back to it.
- I will follow suggested outline
- Try to work problems in team.
  - We all have different strengths and weaknesses

Path

- The only way to success will be
  consistency and dedication
- This is no different that preparing for a Soccer game or a Chess Match.
  - Practice, Practice and Practice.
- Attend Thursday lectures and Friday Practices

Outline of sessions

- Chapter 2
- 1.2 , 3.1 – 3.5
- 4.1 – 4.5
- 3.5, 4.7.1
- 5.4, 5.6
- 6.5, 8.4
- 4.6-4.7.4
- Chapter 5
- Chapter 6
- Chapter 7
- Chapter 8

Final tips

- I’m here to help you … I’m just your coach
  - Don’t expect your coach to win the games for you

Linear Data Structures

- Static Array
  - int[5] myArray = {1,2,3}; // initial values 1,2,3,0,0
- Resizeable Array
  - vector<int> v(5, 5); //initial values 5,5,5,5,5

```
for (int i = 0; i < 5; i++) {
        arr[i] = i;
        v[i] = i;
 }
v.push_back(10);  //v has 0,1,2,3,4,10 now!
```

Sorting in C++

- For the following declaration

```
int *pos, arr[] = {10, 7, 2, 15, 4};
vector<int> v(arr, arr + 5);
vector<int>::iterator j;
```

Sorting in C++ (2)

```
// sort descending with vector
  sort(v.rbegin(), v.rend());  // example 'reverse iterator'
  for (vector<int>::iterator it = v.begin(); it != v.end(); it++)
    printf("%d ", *it);           // access the value of iterator

  // sort descending with integer array
  sort(arr, arr + 5);             // ascending
  reverse(arr, arr + 5);        // then reverse
  for (int i = 0; i < 5; i++)
    printf("%d ", arr[i]);
```

Sorting in C++(3)

```
random_shuffle(v.begin(), v.end());
  for (vector<int>::iterator it = v.begin();
                                        it != v.end(); it++)
    printf("%d ", *it);

partial_sort(v.begin(), v.begin() + 2, v.end());
…
```

Sorting in C++(4)

```
// sort ascending
sort(arr, arr + 5);
 // sorting a vector
sort(v.begin(), v.end());
```

Sorting multi-field in C++

```
typedef struct {
  int id;
  int solved;
  int penalty;
} team;

bool icpc_cmp(team a, team b) {
  if (a.solved != b.solved)
            return a.solved > b.solved;
  else if (a.penalty != b.penalty)      // a.solved == b.solved
            return a.penalty < b.penalty;
   else                        // a.solved == b.solved AND a.penalty == b.penalty
             return a.id < b.id;
}
```

Sorting multi-field in C++(2)

```
// multi-field sorting example, suppose we have 4 ICPC teams
 team nus[4] = { {1, 1, 10},
                 {2, 3, 60},
                 {3, 1, 20},
                 {4, 3, 60} };

 sort(nus, nus + 4, icpc_cmp);  // sort using a comparison func.
 printf("==================\n");
for (int i = 0; i < 4; i++)
   printf("id: %d, solved:%d, penalty: %d\n",
         nus[i].id, nus[i].solved, nus[i].penalty);
```

Sorting multi-field in C++(3)

```
// there is a trick for multi-field sorting if the sort order is "standard"
  // use "chained" pair class in C++ and put the highest priority in front

 typedef pair < int, pair < string, string > > state;
  state a = make_pair(10, make_pair("steven", "grace"));
  state b = make_pair(7, make_pair("steven", "halim"));
  state c = make_pair(7, make_pair("steven", "felix"));
  state d = make_pair(9, make_pair("a", "b"));
  vector<state> test;
  test.push_back(a);
  test.push_back(b);
  test.push_back(c);
  test.push_back(d);
```

Sorting multi-field in C++(4)

sort(test.begin(), test.end());

WHY?

In inequality comparisons (<, >),
the first elements are compared first, and only if the inequality comparison is not true for them, the second elements are compared.

Binary Search in C++

```
pos = lower_bound(arr, arr + 5, 7);                    // found
printf("%d\n", *pos);
j = lower_bound(v.begin(), v.end(), 7);
printf("%d\n", *j);

pos = lower_bound(arr, arr + 5, 77);                   // not found
if (pos - arr == 5) // arr is of size 5 ->
        printf("77 not found\n");

j = lower_bound(v.begin(), v.end(), 77);
if (j == v.end())          //return vector index +1 of vector size
        printf("77 not found\n"); // WHY???? Anyone?
```

Ways to search for an item

- O(n) . Linear Search
- O(log n) Binary Search
- O(1) with Hashing (but can live without it for the contest)

Permutations

next_permutation(arr, arr + 5);
// 2, 4, 7, 10, 15 -> 2, 4, 7, 15, 10
next_permutation(arr, arr + 5);
// 2, 4, 7, 15, 10 -> 2, 4, 10, 7, 15
OR using vectors
next_permutation(v.begin(), v.end());
next_permutation(v.begin(), v.end());

// sometimes these two useful simple macros are used
printf("min(10, 7) = %d\n", min(10, 7));
printf("max(10, 7) = %d\n", max(10, 7));

Linked List

- In most cases, we can use vector.
  - Exception Uva 11988
    - Broken Keyboard (a.k.a Beiju Text)

Stacks

- Only insertion (push) and deletion (pop) from the top only
  - LIFO
- top() : obtain content from the top of the stack
- empty()

Queues

- Used in Bread First Search (BFS)
- Insertion (enqueue) from back
- Deletion (dequeue) from head
  - FIFO
- push(), pop()
- front(),back()
- empty()

Stack and Queue C++ (1)

```
stack<char> s;
s.push('a');
s.push('b');
s.push('c');
// c <- top
// b
// a
printf("%c\n", s.top());          // output 'c'
s.pop();                          // pop topmost
printf("%c\n", s.top());          // output 'b'
printf("%d\n", s.empty());        // currently s is not empty
```

Stack and Queue C++ (2)

```
queue<char> q;
while (!s.empty()) {
   q.push(s.top());      // enqueue 'b', and then 'a'
   s.pop();
 }
 q.push('z');              // add one more item
 printf("%c\n", q.front());    // prints 'b'
 printf("%c\n", q.back());     // prints 'z'
```

BITMASK C++

- Use bitset whenever possible.
  - Faster than array of bits and vector<bool>

```
char bmask[4] = {'1','0','1','1'};
bitset<8> bs(bmask); //11
//bs is stored as 1101
cout << bs.at(2) << endl; //0
cout << bs << endl;
cout << bs.flip() << endl;
```

Some Binary Operations

- Using int (32 bits) makes faster than bitset, vector<bool>…
- S = 34
- S = S << 1 = S * 2
- S = S >> 2 = S / 4
- S = S << 1 = S / 2

Some Binary Operations (2)

- Turn a bit
- 1 << 3 = 1000
- C++ binary operation
  - & = and
  - | = or
  - ^ = XOR
  - ~ = NOT

Some Binary Operations (3)

```
#define isOn(S, j) (S & (1 << j))
#define setBit(S, j) (S |= (1 << j))
#define clearBit(S, j) (S &= ~(1 << j))
#define toggleBit(S, j) (S ^= (1 << j))
#define lowBit(S) (S & (-S))
#define setAll(S, n) (S = (1 << n) - 1)
```

Some Binary Operations (3)

- Look at ch_02_bit_manipulations.cpp

Non-Linear Data Structures C++

- Balanced Binary Search Tree (AVL or RB)
  - C++ stl map/set uses RB
    - Worst case O(log n) for
      - Search
      - Deleteion
      - Insert
- Hash Tables
  - No native STL hash table in C++
    - Java HashMap,HashSet,HashTable
  - Avoid in programming contest or learn how to implement it

Next Session

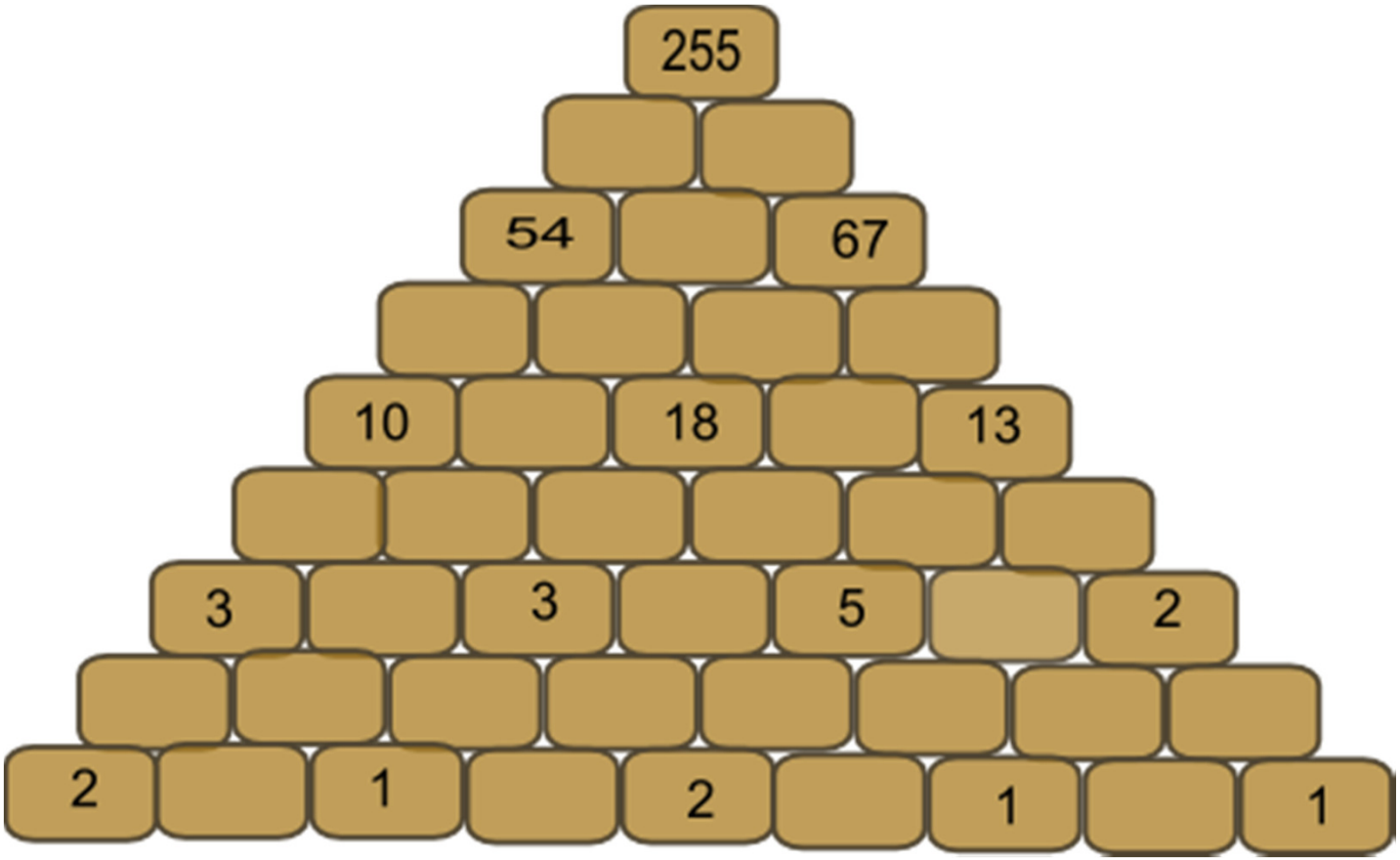
- C++ STL Priority Queue
- Data Structures with Our-Own Libraries (2.3)

Problems to tackle

- Basic Data Structures
  - Uva 11040 Add Bricks in the wall
  - UVa 11340 – Newspaper
  - Uva Grid Successors
- C++ STL Algorithm

Add bricks to the wall Uva 11040

- Uva 11040



Add Bricks to the wall input/output

- **Sample Input**

2 255 54 67 10 18 13 3 3 5 2 2 1 2 1 1 256 64 64 16 16 16 4 4 4 4 1 1 1 1 1

- **Sample Output**

255 121 134 54 67 67 23 31 36 31 10 13 18 18 13 5 5 8 10 8 5 3 2 3 5 5 3 2 2 1 1 2 3 2 1 1 2 0 1 0 2 1 1 0 1 256 128 128 64 64 64 32 32 32 32 16 16 16 16 16 8 8 8 8 8 8 4 4 4 4 4 4 4 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1

Another brick to the wall HINTS

- Recursive
- Wall = W [ROW] [COLUMN]
- W[2][1] = (W[0][0] – W[2][0] – W[2][2]) / 2
- W[1][0] = (W[2][0] + W[2][1])
- W [1][1] = (W[2][1] + W[2][2])

Splitting Numbers Uva 11933

We define the operation of splitting a binary number n into two numbers $a(n)$, $b(n)$ as follows. Let $0 i_1 < i_2 < \ldots < i_k$ be the indices of the bits (with the least significant bit having index 0) in n that are 1. Then the indices of the bits of $a(n)$ that are 1 are $i_1, i_3, i_5, \ldots$ and the indices of the bits of $b(n)$ that are 1 are $i_2, i_4, i_6, \ldots$

HINT: Bit Manipulation

HINT: Use 2 Mask with &

Splitting Numbers Input/Output

- **Sample Input**
- 6 7 13 0
- **Sample Output**
- 2 4 5 2 9 4

Newspaper Uva 11340

- News agency pays money for articles according to some rules. Each character has its own value (some characters may have value equals to zero). Author gets his payment as a sum of all character's values in the article. You have to determine the amount of money that news agency must pay to an author.
- HINT: DIRECT ADDRESSING TABLE

Newspapwer Input/Output

- **SAMPLE INPUT:**

1 7 a 3 W 10 A 100 , 10 k 7 . 3 l 13 7
ACM International Collegiate Programming Contest (abbreviated as ACM-ICPC or just ICPC) is an annual multi-tiered competition among the universities of the world. The ICPC challenges students to set ever higher standards of excellence for themselves through competition that rewards team work, problem analysis, and rapid software development. From Wikipedia.

- **SAMPLE OUTPUT:**

3.74$

Other Problems to tackle

- C++ Stl Algorithms (Java Collections)
  - UVa 10194 Soccer
  - UVa 11588 Image Coding
  - UVa 00146 ID Codes
- Sorting
  - UVa 00855 Lunch in the Grid City
- Stacks
  - Uva 00514 Rails
- Queue
  - Uva 10901 Ferry Loading III