Announcements

- TA video for homework will be published in classroom every week
- We will have midterm next week
- 5 problems, 3 regular, 100 pt each, 2 bonus 10 pt each
- Scores will be counted from high to low
- E.g. if you get 100, 50, 100, 90, 10 in XJOI, your grade will be 100 + 100 + 90+ 5 + 1 = 296/300
- Do not open test page unless you are ready for the test. Timer will start immediately

Catch the cow (P8115)

*Walking: John can move from any point X to X-1 or X+1 point in one minute

*Transport: John can move from any point X to point 2*X in one minute.

Sample input:

5 17

Sample output:

4

Hint

Solution for sample data is: 5-10-9-18-17. Thus it needs 4 minutes by minium.

Catch the cow (P8115)

```
13  // not empty
14  while( !que.empty()) {
15    //pop front
16    int cur = que.front();
17    if (cur == target) {
18      return dis[cur];
19    }
20    que.pop();
21
22    // traverse
23    if ( cur + 1 < 1000001 && !dis[cur+1]) {
24      dis[cur + 1] = dis[cur] + 1;
25      que.push(cur+1);
26    }
27    if (cur - 1 >= 0 && !dis[cur-1]) {
28      dis[cur - 1] = dis[cur] + 1;
29      que.push(cur-1);
30    }
31    if (cur * 2 < 1000001 && !dis[cur*2]) {
32      dis[ cur *2] = dis[cur] + 1;
33      que.push(cur*2);
34    }
35  }
36
37  return dis[target];
```

```
40  int main() {
41    int start;
42    cin >> start >> target;
43    memset(arr, 0, sizeof(arr));
44    memset(dis, 0, sizeof(dis));
45
46    cout <<  bfs(start) << endl;
47  }
```

Little fish (P7670)

A little fish needs to remember a row of numbers he saw (The length of the row is not fixed and ends with "0". The maximum row length is no more than 1000000, and the number is no more than 2 ^ 32-1). He then needs to read out them in the reverse order (the number 0 for the (original) end should not be read out). It's too hard for the little fish to remember. You don't want to think about how big his brain is. So please help out the fish with a program.

Sample Input:

3 65 23 5 34 1 30 0

Sample Output:

30 1 34 5 23 65 3

Little fish (P7670)

```cpp
int main() {
  int num;
  stack<int> stac;
  while(cin >> num && num) {
    stac.push(num);
  }

  while(!stac.empty()) {
    cout << stac.top() << " ";
    stac.pop();
  }
}
```

Parentheses (P9428)

Given an expression string exp consisting of parentheses characters "(" and ")", bracket characters "[" and "]", and brace characters "{" and }", check whether the pairs and the orders of these characters are matched in expr.

Examples:

expr = "([]())", output: OK

expr = "[({}[])]", output: OK

exp = “[()]{}{[()()]()}”, output: OK

exp = “[(])”, output: Wrong

expr = "([]()", output: Wrong

expr = "(()))", output: Wrong

Parentheses (P9428)

```
25 ▼  for(int i = 0; i < inp.size(); i++) {
26      //left, push to stack
27 ▼    if(inp[i] == '(' || inp[i] == '{' || inp[i] == '[') {
28        stac.push(inp[i]);
29      }
30      // else, try to pop
31 ▼    else {
32        // check empty
33 ▼      if(!stac.empty()) {
34          // check if expression matches
35 ▼        if(check_match(inp[i], stac)) {
36            stac.pop();
37          }
38          // 1. not match
39 ▼        else {
40            cout << "Wrong" << endl;
41            return 1;
42          }
43        }
```

```
44            //3. stack is empty
45 ▼        else {
46            cout << "Wrong" << endl;
47            return 1;
48          }
49        }
50      }
51      //2. stack is not empty at the end
52 ▼    if(!stac.empty()) {
53        cout << "Wrong" << endl;
54      }
55 ▼    else {
56        cout << "OK" << endl;
57      }
```

Mathematical expression (P1321)

Calculate the value of a mathematical expression with a length not exceeding 100. All intermediate operations will within the range of the basic type; the operation rules are parentheses, powers, multiplications, divisions, additions and subtractions,from top to bottom,and in the same level ,from left to right. The result is rounded off to retain 2 decimal places.

Sample input:

2^3+5/3*(231-130)+10000

Sample output:

10,176.33

Mathematical expression (P1321)

-5+2^3+5/3*(231-130)+123.34-4

```
5   char s[1000];
6   int  g_pos;  // current index of string
7
8   // read whole number
9   double Translation(int & pos)
10  ▶{ … }
37
38  // priority level
39  int GetLevel(char ch)
40  ▶{ … }
58
59  // calculate two numbers and an operator
60  double Operate(double a1, char op, double a2)
61  ▶{ … }
77
78  // main calculation function
79  double Compute()
80  ▶{ … }
169
170 int main()
171 ▼{
172     cin>>s;
173     printf( "%.2lf" ,  Compute() ) ;
174
175     return 0;
176 }
```

```
8   // read whole number
9   double Translation(int & pos)
10  ▼{
11      double integer = 0.0;    // integer part
12      double remainder = 0.0;  // remainder part
13
14      while (s[pos] >= '0' && s[pos] <= '9')
15      {
16          integer *= 10;
17          integer += (s[pos] - '0');
18          pos++;
19      }
20
21      if (s[pos] == '.')
22      {
23          pos++;
24          int c = 1;
25          while (s[pos] >= '0' && s[pos] <= '9')
26          {
27              double t = s[pos] - '0';
28              t *= pow(0.1, c);
29              c++;
30              remainder += t;
31              pos++;
32          }
33      }
34
35      return integer + remainder;
36  }
```

```
39  int GetLevel(char ch)
40  {
41      switch (ch)
42      {
43      case '+':
44      case '-':
45          return 1;
46      case '*':
47      case '/':
48          return 2;
49      case '(':
50          return 0;
51      case '#':
52          return -1;
53      case '^':
54          return 3;
55      };
56      return 1;
57  }
58
59  // calculate two numbers and an operator
60  double Operate(double a1, char op, double a2)
61  {
62      switch (op)
63      {
64      case '+':
65          return a1 + a2;
66      case '-':
67          return a1 - a2;
68      case '*':
69          return a1 * a2;
70      case '/':
71          return a1 / a2;
72      case'^':
73        return  pow(a1,a2);
74      }
75      return 0.0;
76  }
```

Mathematical expression (P1321)

```
double Compute()
{
    stack<char> optr;    // for operators
    stack<double> opnd;  // for operands

    optr.push('#');      // help us to know end of calculaiton
    int len = strlen(s);
    bool is_minus = true;  // minus sign or negative sign?
                           // e.g. -5+3*2 or 3+(-2)+6

    for (g_pos = 0; g_pos < len;)
    {
        //1. check minus sign
        if (s[g_pos] == '-' && is_minus)  // minus sign
        {
            opnd.push(0); // add a 0
            optr.push('-');
            g_pos++;
        }
        //2. closing parenthese ')'
        else if (s[g_pos] == ')')
        {
            is_minus = false;
            g_pos++;

            while (optr.top() != '(')
            {
                double a2 = opnd.top();
                opnd.pop();
                double a1 = opnd.top();
                opnd.pop();
                char op = optr.top();
                optr.pop();

                double result = Operate(a1, op, a2);
                opnd.push(result);
            }
            optr.pop();  // delete '('
        }
        //3. this is a number
        else if (s[g_pos] >= '0' && s[g_pos] <= '9')
        {
            is_minus = false;
            opnd.push(Translation(g_pos));
        }
        //4. opening parenthese
        else if (s[g_pos] == '(')
        {
            is_minus = true;
            optr.push(s[g_pos]);
            g_pos++;
        }
        //5. this is an operator
        else
        {
            while (GetLevel(s[g_pos]) <= GetLevel(optr.top()))
            {
                // current priority ls lower than the top of stack
                // finish the calculation using top of stack
                double a2 = opnd.top();
                opnd.pop();
                double a1 = opnd.top();
                opnd.pop();
                char op = optr.top();
                optr.pop();

                double result = Operate(a1, op, a2);
                opnd.push(result);
            }

            optr.push(s[g_pos]);
            g_pos++;
        }
    }

    while (optr.top() != '#')
    {
        double a2 = opnd.top();
        opnd.pop();
        double a1 = opnd.top();
        opnd.pop();
        char op = optr.top();
        optr.pop();

        double result = Operate(a1, op, a2);
        opnd.push(result);
    }

    return opnd.top();
}
```

X - Camp

Set & Map

STL standard container

In this course, we will learn the use of set and map

Class : 202

SET

std::set is the container that store unique elements following a specific order.

Set properties:

- Each value is unique, even if you try to insert multiple times.
- Each element can be easily inserted or removed by value or iterator
- The elements in the set is automatically ordered (<)
- The value of an element also identifies it, so you can use find(val)
- The value of the elements in a set cannot be modified once in the container

SET

std::set

Definition and initialization:

```
set<int> a; // define a set a of type int
set<int> b(a); // define and initialize set b with set a
set<int> b(a.begin(), a.end()); // Use all elements in set a as the initial value of set b

int n[] = { 1, 2, 3, 4, 5 };
set<int> a(n, n + 5); // Use the first 5 elements of array n as the initial value of set a
```

SET

Now let's look at the CRUD operation of set

C(Create):

- `st.insert(const T& x);//Insert an element into the container:`

```
#include <iostream>
#include <set>
using namespace std;
int main(){
    set<int> st;
    st.insert(4);            // insert the element into the container
    st.insert(2);
    //
    for (auto it = st.begin(); it != st.end(); it++)
        cout << *it << " ";      // 2 4
    cout << endl;
    return 0;
}
```

SET

- D(Delete):

- Remove the element whose value is elem in the container: `st.erase(const T& elem);`
- Delete the element pointed to by the it iterator: `st.erase(iterator it);`
- Delete all elements between the interval [first, last]: `st.erase(iterator first, iterator last);`
- Clear all elements: `st.clear();`

```cpp
#include <iostream>
#include <set>
using namespace std;
int main(int argc, char* argv[]){
    set<int> st;
    for (int i = 0; i < 8; i++)
    st.insert(i);
    // delete the element whose value is elem in the container
    st.erase(4);
    // delete an element anywhere
    set<int>::iterator it = st.begin();
    st.erase(it);
    // delete elements between [first, last]
    st.erase(st.begin(), ++st.begin());
    // traverse the display
    for (it = st.begin(); it != st.end(); it++)
        cout << *it << " "; // output: 2 3 5 6 7
    cout << endl;
    // clear all elements
    st.clear();

    // Check if set is empty
    if (st.empty())
        cout << "collection is empty" << endl; // output: collection is empty
    return 0;
}
```

SET

R(Retrieve):

```
st.find(key);//Find if the key key exists
// Returns an iterator over the elements of the key if it exists;
// returns set.end() if it does not exist
// or st.count(key) (0 or 1 to indicate existence)

#include <iostream>
#include <set>
using namespace std;
int main(){
    set<int> st;
    for (int i = 0; i < 6; i++)
    st.insert(i);
    // Find the key value by find(key)
    set<int>::iterator it;
    it = st.find(2);
    cout << *it << endl; // output: 2
    return 0;
}
```

SET

- U(Update):

- The iterator of the set does not overload the "=" symbol

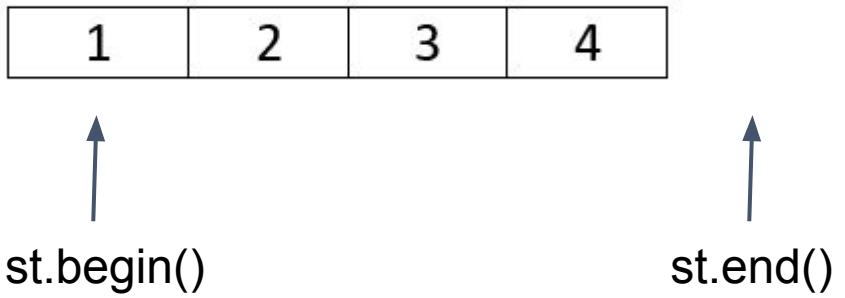
To put it bluntly, that is, the compiler imposes restrictions and does not allow us to modify the content of the set iterator.

*it = 0; // this is wrong

iterator

- Common iterator functions

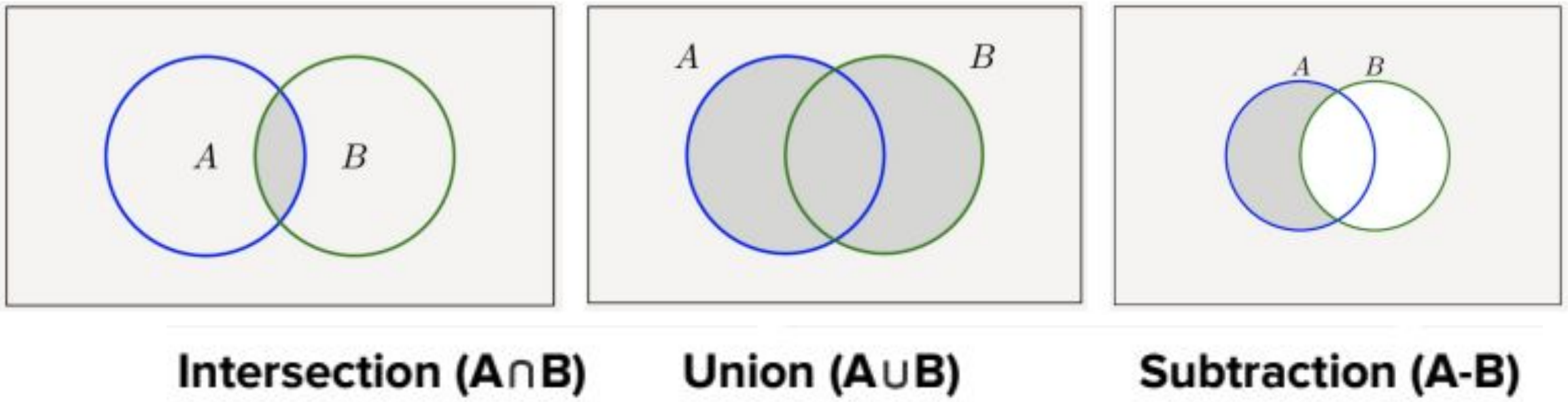
```
Start iterator pointer: st.begin();

End iterator pointer: st.end(); // points to the next position of the last element

Reverse iterator pointer, pointing to the last element: st.rbegin();

Reverse iterator pointer to the previous element of the first element: st.rend();
```



SET

set<int> s declare an int set s
set<int>::iterator it declare set iterator it
s.begin() the beginning iterator
s.end() the end iterator (after the last element)
s.insert(a) insert element a into the set s, keeping the order.
s.find(a) find element a in the set s and return the iterator; return s.end() if not found
s.count(a) find element a in the set a and return count (0 or 1 in case of set)
s.erase(a) erase the element a from the set s
s.clear() clear the set s
s.size() return the size of set s
s.empty() true or false

set operations



Please output the union and intersection of the following 2 sets
9 3 7 6 5
10 5 4 3 2 7

set of struct

operator< needs to be defined for the element order

```
struct point {
	int x, y;
	point(int a, int b) : x(a), y(b) {};
};
bool operator< (point a, point b) {
	return (a.x < b.x || (a.x == b.x && a.y<b.y));
}

set<point> ps;
ps.insert(point(2, 8));
ps.insert(point(1, 2));
ps.insert(point(1, 3));
set<point>::iterator it = ps.find(point(1,2));
```

Exercise: set of struct

Enter coordinates of n points, output the point closest to (0, 0)

```
struct point {
    int x, y;
    point(int a, int b) : x(a), y(b) {};
};
bool operator< (point a, point b) {
return (a.x*a.x + a.y*a.y) < (b.x*b.x+b.y*b.y);
}

set<point> ps;
ps.insert(point(2, 8));
ps.insert(point(1, 2));
ps.insert(point(1, 3));
ps.insert(point(2, 0));
```

set of struct

Enter coordinates of n points, output the point closest to (0, 0)

```
struct point {
    int x, y;
    point(int a, int b) : x(a), y(b) {};
};
bool operator< (point a, point b) {
    return (a.x*a.x + a.y*a.y) < (b.x*.b.x+b.y*b.y);
}

set<point> ps;
ps.insert(point(2, 8));
ps.insert(point(1, 2));
ps.insert(point(1, 3));
set<point>::iterator it = ps.begin();
cout << (*it).x << " " << (*it).y << endl;
```

map

Many data in reality are related, such as book name and price. Each piece of data contains two parts:

Informatics Competition: ¥80
Advanced Mathematics: ¥27.5
Biology : ¥35.5

We can use map to store this kind of one-to-one data:

The first one can be called a key, and each key can only appear once in the map;
The second may be called the value of the keyword;

map

Initialization

```
std::map<std::string, int>myMap;
myMap["C language tutorial"] = 10;
std::map<std::string, int>myMap{ {"C language tutorial",10},{"STL tutorial",20} };
//myMap container contains 2 key values in the initial state right.
```

map

- Insert element:

- The [ ] operator is overloaded in the C++ STL map class template, that is, according to different usage scenarios, different operations can be achieved with the [ ] operator. for example

```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){
    std::map<string, string> mymap{ {"C++","printf,scanf"} };
    //Get the value corresponding to the specified key in the stored key-value pair
    cout << mymap["STL"] << endl;

    //Add a new key-value pair to the map container
    mymap["Python"] = "print,input";

    //Modify the value corresponding to the specified key of the map container
    mymap["C++"] = "cout,cin";

    for (auto iter = mymap.begin(); iter != mymap.end(); ++iter) {
        cout << iter->first << " " << iter->second << endl;
    }
    return 0;
}
```

map

- get element

- The [ ] operator is overloaded in the map class template, which means that, similar to the direct access to the elements in the array with the help of the array subscript, we can easily obtain the value corresponding to the key in the map container through the specified key.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    //Create and initialize the map container
    std::map<std::string, std::string>myMap{ {"C++","printf"},
                                             {"Python","print"},
                                             {"Java","System.out.print"} };
    string cValue = myMap["C++"];
    cout << cValue << endl;
    return 0;
}
```

map

- iterate over elements

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ////Create and initialize the map container
    map<string, string>myMap{ {"C++","printf"},{"Python","print"} };

    //Call the begin()/end() combination to traverse the map container
    for (map<string, string>::iterator iter = myMap.begin(); iter != myMap.end(); ++iter) {
        cout << iter->first << " " << iter->second << endl;
    }
    return 0;
}
```

map

- // delete element

```
//delete the element pointed to by the key bfff

cmap.erase("bfff");


// delete the element pointed to by the iterator key
map<string,int>::iterator key = cmap.find("mykey");
if(key!=cmap.end()){
     cmap.erase(key);
}

// delete all elements
cmap.erase(cmap.begin(),cmap.end());
```

map

```
map<int, string> mp; declare an int map
map<int, string>::iterator it; declare map iterator it
mp.begin() the beginning iterator
mp.end() the end iterator (after the last element)
mp[5] = “red”; assign value to a key
mp.count(5) find if a key in map (return 0 or 1)
mp.erase(5) erase the key (and its value)
mp.clear() clear the map
mp.size() return the size
mp.empty() true or false
```

Map of struct

You can use struct as key of map, but need to define < operator

```
struct point {
  int x, y;
  point() : x(0), y(0) {}; // better to have default constructor, equivalent to “point() { x=0; y=0;}”
  point(int a, int b) : x(a), y(b) {};  // equivalent to “point(int a, int b) { x=a; y=b;}”
};
bool operator< (point a, point b) {
  return (a.x < b.x || (a.x == b.x && a.y<b.y));
}

int main() {
map<point, bool> visited;
point p1(1, 3); point p2(0, 0);
map<point, bool>::iterator it;
visited[p1]=false; visited[p2]=true;
for (it = visited.begin(); it != visited.end(); it++)
    cout << (*it).first.x << "," << (*it).first.y << " is " << (*it).second << endl;
  }
```

Map - exercise

- Create a struct “student” for parameters “first, last, midterm”
- Create a map: key - studentID, value - student struct
- Insert the elements {“xc1005, John, Li, 98”, “xc1001, Mike, Wang, 89”, “xc1003, John, Li, 78”, “xc1009, Tom, Chen, 88”}
- There are two “John Li” but with different ID
- Output all students in the map for ID, first name, last name, and midterm score. Predict what output it should be
- Sample output:

  xc1001: Mike Wang 89

  xc1003: John Li 78

  xc1005: John Li 98

  xc1009: Tom Chen 88

Map - exercise

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  struct student {
5      string firstName, lastName;
6      int midterm;
7      student() : firstName(""), lastName(""), midterm(0) {};
8      student(string f, string l, int m) :
9          firstName(f), lastName(l), midterm(m) {};
10 };
11
12 int main() {
13     map<string, student> mymap;
14     mymap["xc1005"] = student("John", "Li", 98);
15     mymap["xc1003"] = student("John", "Li", 78);
16     mymap["xc1001"] = student("Mike", "Wang", 89);
17     mymap["xc1009"] = student("Tom", "Chen", 88);
18
19     for(auto it = mymap.begin(); it != mymap.end(); it++) {
20         cout << it->first << ": " << it->second.firstName << " " <<
21            it->second.lastName << " " << it->second.midterm << endl;
22     }
23 }
```

What we learned so far?

- Reviewed basic DFS and BFS, floodfill, shortest path
- New applications for DFS/BFS
  - Representation of node
  - state transition
  - check availability
- New C++ concepts
  - scanf/printf
- New data structures
  - struct, queue, stack, set, map

How to prepare for midterm

- Finish all missing homework
- Review class slides/recording/homework
- No need to memorize C++ syntax/system functions. You are allowed to search if you forget