NITK Codechef Campus Chapter

COMPETITVE PROGRAMMING LECTURE SERIES
TALK 1

TOPICS : STL & IMPLEMENTATION

By : R Anirudh

Main Building
February 8th 2014

Why STL ?

- Standard Template Library is poweful tool for effecient programming.
- It reduces length of the code and makes it clean.
- It is well tested and optimized.
- Advantage over your peers. If you know STL you sort an array with a line of code while your friends write the entire nlogn algorithm such as mergesort for the same.

PRE-REQUISITES TO USE STL

- Add #include<algorithm> to use the library
- Add other header files such as #include<stack> for stack data structure. Similarly for other data-structures.
- Use a good reference material . You need not remember every function. Knowing how to use a reference material is suuficient :)

Links for materials

1. http://www.cplusplus.com/reference
2. https://www.sgi.com/tech/stl/

USE OF STL IN ALGORITHMS

Terminology in STL

- A range is any sequence of objects that can be accessed through iterators or pointers, such as an array or an instance of some of the STL containers.

  Syntax :-

  STL_FUNC(begin iterator,end iterator)

- Assume an array called 'arr' with 'n' elements for the entire lecture.

*Iterator and pointer may be used interchangbly

SIMPLE UTILITY FUNCTIONS

- Swapping two nos. a & b

  swap(a,b)

  Time complexity – constant

- Minimum of two nos. a &b

  min(a,b)

  Time complexity – Constant

- Similarly for maximum use max(a,b)

UTILITY FUNCTIONS IN ARRAYS

- To find the maximum value in an array of n elements simply use

  max_element(arr,arr+n)

- Similarly for minimum use

  min_element(arr,arr+n)

- To sort an array use the function sort

  sort(arr,arr+n)

  Time complexity – O(nlogn) is as effecient as the commonly used merge/heap sort.

UTILITY FUNCTIONS IN ARRAYS

- To reverse the the order of the elements in the range [first,last) use

  reverse(arr,arr+n)

  Complexity – Linear

- To find an element in a given range use

  find(arr,arr+n)

  Returns an iterator to the first element in the range [first,last) that compares equal to val. If no such element is found, the function returns last.

  Time Complexity - Linear

OTHER USEFUL FUNCTIONS

- binary_search(arr,arr+n,search_value) : Test if value exists in sorted sequence
- count(arr,arr+n,val) : Returns the number of elements in the range [first,last) that compare equal to val.
- lower_bound(arr,arr+n,val) : Returns an iterator pointing to the first element in the range [first,last) which does not compare less than val
- upper_bound(arr,arr+n,val) : Returns an iterator pointing to the first element in the range [first,last) which compares greater than val.
- next_permuatation(arr,arr+n) : Rearranges the elements in the range [first,last) into the next lexicographically greater permutation.

STL and Data-Structures

Vector

- The simplest container. It is similar to an array with additional features. Infact, the vector library has been built using arrays.
- Declare a vector

```
vector<int> v;
```

- vector<int> v(10) means an array with 10 elements. You have allocated the memory before hand for the vector and cannot assign any more lements.
- Default values in a vector is 0.
- If you wish to add elements to your declaration vector<int> v then use the function

```
v.push_back(val);
```

  This adds an element at the back of the vector. This is an example of dynamic memory allocation.

- To get the size of a vector use the function : v.size()

Vector

- To check if a vector is empty use function v.empty() which returns true/false depending on the content of the vector
- Suppose you want to resize your vector declared as vector<int> v(10) to 15 elements use v.resize(15) . The resize() function makes vector contain the required number of elements. If you require less elements than vector already contain, the last ones will be deleted.
- To clear a vector use v.clear()
- To initialize a vector from another vector there are two ways

```
vector<int> v1;
vector<int> v2 = v1;
vector<int> v3(v1);
int arr[]={1,2,3};
vector<int> v(arr,arr+sizeof(arr)/sizeof(int));
```

Vector

- Creating multi-dimensional vectors. Can be done by using a vector of vectors

  ```
  vector<vector<int> > matrix
  ```

- It should now be clear how to create a 2-d vector of n*m

  ```
  vector<vector<int> > matrix(n,vector<int>(m))
  ```

  To initiliaze the same 2-d vector with a value

  ```
  vector<vector<int> > matrix(n,vector<int>(m,55))
  ```

- For more functions and features refer to STL guides and use google to seek answers to your doubts. Most are available on stackoverflow.

Pairs

- An important data-structure resembles a structure of two-elements .

  pair<int,int> pii;

  How to create a pair ?

  Pii = make_pair(10,15)
- The great advantage of pairs is that they have built-in operations to compare themselves . To access pairs use

  int x = pii.first;

  int y = pii.second;
- Pairs become very important in sorting by value.

Strings

- STL has in-built container to manipulate strings.
- Resembles java strings.
- Makes string functions very simple.

If you want to concatenate two strings

string c = string a + string b

Sets

Used when ?

- add an element, but do not allow duplicates.
- Remove elements
- Get a count of distinct elements

Sets

Implementaion

```
set<int> s;
for(int i = 1; i <= 100; i++) {
  s.insert(i); // Insert 100 elements, [1..100]
}
s.insert(5);    // Doesn't do anything. Duplicate !!
s.erase(6);   // Removes 6th element
```

Maps

- Maps contain pairs<key,value>
- Map ensures that at most one pair with specific key exists

```
map<string, int> M;
 M["Top"] = 1;
 M["Coder"] = 2;
 M["SRM"] = 10;

 int x = M["Top"] + M["Coder"];
```

Maps

- Iterating through a map.

```
map<int,string> m;
m[1]=”Ajith”;
m[2]=”Kumar”;
map<int,string>::iterator it;
for(it=m.begin();it!=m.end();it++)
{
   if(it->second=="ajith”)
       cout<<”Here's our superstar”;
}
```

Stacks

- Last in , first out (LIFO)
- Supports three constant-time operations

  - push(x) : inserts x into the stack
  - pop(x) : removes the newest element
  - top() : returns the topmost element.

Stacks

- Declare #include<stack>
- Create a stack by simply writing

  stack<data_type> name_of_stack

  For example : stack<int> mystack
- To push an element

  mystack.push(x);
- To pop an element

  mystack.pop();
- To get the top of the stack

  mystack.top();
- To check if the stack is empty (crucial in case of under-flow !!!)

  mystack.empty() returns true/false depending on the content of the stack.

Problems on Stacks

- http://codeforces.com/problemset/problem/344/D
- http://www.codechef.com/problems/BEX

Problems on Stacks

- http://codeforces.com/problemset/problem/344/D
- http://www.codechef.com/problems/BEX
- http://www.spoj.com/problems/STPAR/

Queues

- First in , first out (LIFO)
- Supports three constant-time operations
    - Enque(x) : inserts x into the stack
    - Dequeue(x) : removes the oldest element
    - front() : returns the oldest item.

Queues

```
queue<int> Queue;
Queue.push(5);
Queue.push(6);     // Inserts an element into a queue
int x = Q.front();     // Returns the oldest element
Queue.pop();     // Pops the oldest element
Queue.empty();   // Returns true/false depending on content
```

Importance of STL

- Makes it very easy to implement graph algorithms like Dijkstra's Shortest Path Algorithm by making use of priority_queue or else we would have to implement a heap from the scratch.
- Min-heaps and max-heaps are easy to use because of the priority_queue
- Network Flow algorithms & graphs are the areas where STL enhances quick and clean coding. We will discuss more features and uses of STL in the upcoming lectures.

Problems on STL

- Set - http://www.spoj.com/problems/FACEFRND
- Sorting based on Pair values - http://www.codechef.com/problems/LEMUSIC
- Maps extensively used - http://www.codechef.com/problems/TOURMAP
- STL problems

1.http://www.spoj.com/problems/AMR12G/

2.http://www.spoj.com/problems/HOMO/

3.http://www.spoj.com/problems/SANTA1/

Resources

- Topcoder Tutorial on STL : http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=standardTemplateLibrary
- C++ Reference – http://cplusplus.com/reference
- Code chef – http://codechef.com
- SPOJ – http://spoj.com
- Code forces – http://codeforces.com

Need more help ?

- Try reading the topcoder tutorial thourougly + go through the STL manuals suggested.
- Solve a good numbers of problems suggested here.
- Do try SPOJ problems from 1-100 sorted in order of increasing order of difficulty

  http://www.spoj.com/problems/classical/sort=-6

If you still have doubts/queries feel free to contact

Anirudh – anirudht20@gmail.com

Tushar – tusharmakkar08@gmail.com

Aditya kadam – adityak93@gmail.com

Ashish Kedia – ashish1294@gmail.com