



HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

The background is a solid dark blue color. Overlaid on this is a pattern of small red dots. These dots are arranged to form a large, stylized letter 'A' that is centered on the page. The dots are more densely packed in the center of the 'A' and become sparser towards the edges, creating a gradient effect. The overall composition is clean and modern.

APPLIED ALGORITHMS



ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

APPLIED ALGORITHMS

General introduction
Data structure library

ONE LOVE. ONE FUTURE.

CONTENTS

- Introduction
- Standard Template Library (STL) in C++



INTRODUCTION

- Objective of the subject
 - Study some advanced data structures and algorithms
 - Apply effective algorithms and data structures to solve complex computational problems
 - Analyze the effectiveness of the algorithm
 - Practice algorithmic programming skills
- Practice
 - Programming to solve applied computational problems
 - Submit source code to the automatic scoring system through test cases
 - Each exercise will be described in detail about the problem statement, format of input data and output results

INTRODUCTION

- Subjects
 - Backtracking, branching and bounding
 - Data structures: stack, queue, set, disjoint set, priority queue, segment tree
 - Cumulative array technique (kỹ thuật mảng cộng dồn), 2 pointer technique, bit representation and processing
 - Greedy algorithm, divide and conquer, dynamic programming
 - Algorithms on graphs: DFS, BFS, Strongly Connected Components, Shortest Path, Minimum Spanning Tree, Max-Flow, Max-Matching

INTRODUCTION - Example (P.01.01.01)

- Given 2 integers a and b, calculate sum of them.
- **Data**
 - Line 1 consists of 2 integers a and b ($0 \leq a, b \leq 10^{19}$)
- **Result**
 - Write a number which is them sum of a and b

Stdin	Stdout
3 5	8

INTRODUCTION - Example (P.01.01.01)

- Given 2 integers a and b, calculate sum of them.
- **Data**
 - One line consists of 2 integers a and b ($0 \leq a, b \leq 10^{19}$)
- **Result**
 - Write a number which is them sum of a and b

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a,b;
    cin >> a >> b;
    int res = a + b;
    cout << res;
    return 0;
}
```


INTRODUCTION - Example (P.01.01.01)

- Given 2 integers a and b, calculate sum of them.
- **Data**
 - One line consists of 2 integers a and b ($0 \leq a, b \leq 10^{19}$)
- **Result**
 - Write a number which is them sum of a and b

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a,b;
    cin >> a >> b;
    int res = a + b;
    cout << res;
    return 0;
}
```

Overflow when a and b are large numbers

INTRODUCTION - Example (P.01.01.01)

- Given 2 integers a and b, calculate sum of them.
- **Data**
 - One line consists of 2 integers a and b ($0 \leq a, b \leq 10^{19}$)
- **Result**
 - Write a number which is them sum of a and b

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    unsigned long long a,b;
    cin >> a >> b;
    unsigned long long res = a + b;
    cout << res;
    return 0;
}
```

INTRODUCTION - Example (P.01.01.01)

- Given 2 integers a and b, calculate sum of them.
- **Data**
 - One line consists of 2 integers a and b ($0 \leq a, b \leq 10^{19}$)
- **Result**
 - Write a number which is them sum of a and b

a = 10000000000000000000, b = 10000000000000000000
→ res = 1553255926290448384 (still overflow)

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    unsigned long long a,b;
    cin >> a >> b;
    unsigned long long res = a + b;
    cout << res;
    return 0;
}
```

INTRODUCTION - Example (P.01.01.01)

- Given 2 integers a and b, calculate sum of them.
- **Data**
 - One line consists of 2 integers a and b ($0 \leq a, b \leq 10^{19}$)
- **Result**
 - Write a number which is them sum of a and b

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    unsigned long long a,b, a1, b1,a2,b2;
    cin >> a >> b;
    a1=a/10;    b1=b/10;
    a2 = a%10;    b2 = b%10;
    unsigned long long c1 = a1+b1+(a2+b2)/10;
    unsigned long long c2 = (a2+b2)%10;
    if (c1 > 0) cout << c1 << c2;
    else cout << c2;
    return 0;
}
```

SOLVED!!

Standard Template Library (STL) in C++

- Standard Template Library (STL) in C++
 - Vector, List
 - String
 - Stack, Queue
 - Set
 - Map
 - Priority queue

Standard Template Library (STL) in C++

- Dynamic array
 - Linear array
 - Operations: access, add, remove elements

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    vector<int> V;
    V.push_back(1); V.push_back(2);
    for(int i = 3; i <= 10; i++) V.push_back(i);
    for(int i = 0; i < V.size(); i++) cout << V[i] << " ";
    cout << endl;
    V.erase(V.begin(), V.begin() + 3);
    for(int i = 0; i < V.size(); i++) cout << V[i] << " ";
    cout << endl;
}
```



```
1 2 3 4 5 6 7 8 9 10
4 5 6 7 8 9 10
```

Standard Template Library (STL) in C++ - List

- Doubly linked list
 - Linear data structure
 - Operations: add elements to the beginning, the end, after a position, remove an element from the list

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    list<int> L;
    for(int v = 1; v <= 5; v++) L.push_back(v);
    list<int>::iterator p;
    p = L.begin();
    advance(p,2);
    L.insert(p,2,6);//insert 2 occurrences of 6 after position p
    for(p = L.begin(); p!= L.end(); p++) cout << *p << " ";
}
```



1 2 6 6 3 4 5

Standard Template Library (STL) in C++ - List

- Doubly linked list
 - Linear data structure
 - Operations: add elements to the beginning, the end, after a position, remove an element from the list

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    list<int> L;
    for(int v = 1; v <= 5; v++) L.push_back(v);
    list<int>::iterator p;
    p = L.begin(); advance(p,2);
    cout << "item at position p is " << *p << endl;
    L.erase(p); //remove the item at position p
    for(p = L.begin(); p != L.end(); p++) cout << *p << " ";
}
```



```
item at position p is 3
1 2 4 5
```


Standard Template Library (STL) in C++ - String

- Represents a string of characters
- Operations: assign, concatenate strings, replace substrings, extract substrings, etc.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s1 = "hello";
    string s2 = s1 + " world";
    cout << "s1 = " << s1 << ", s2 = " << s2 << endl;
    string ss = s2.substr(2,6);
    cout << "s2 = " << s2 << ", length = " << s2.length() << endl;
    cout << "s2.substring(2,6) = " << ss << endl;
    s2.replace(6, 5, "abc");
    cout << "new s2 = " << s2 << endl;
}
```



```
s1 = hello, s2 = hello world
s2 = hello world, length = 11
s2.substring(2,6) = llo wo
new s2 = hello abc
```

Standard Template Library (STL) in C++ - Stack

- Linear data structure
- Operations: Add and remove elements with the Last In First Out (LIFO) principle

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    stack<int> S;
    for(int i = 1; i <= 5; i++){
        S.push(i);  cout << "PUSH " << i << endl;
    }
    while(!S.empty()){
        int e = S.top(); S.pop(); cout << "POP " << e << endl;
    }
}
```



```
PUSH 1
PUSH 2
PUSH 3
PUSH 4
PUSH 5
POP 5
POP 4
POP 3
POP 2
POP 1
```

Standard Template Library (STL) in C++ - Queue

- Linear data structure
- Operations: Add and remove elements with the First In First Out (FIFO) principle

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    queue<int> Q;
    for(int e = 1; e <= 5; e++){
        Q.push(e); cout << "Queue push " << e << endl;
    }
    while(!Q.empty()){
        int e = Q.front(); Q.pop(); cout << "Queue POP " << e << endl;
    }
}
```



```
Queue push 1
Queue push 2
Queue push 3
Queue push 4
Queue push 5
Queue POP 1
Queue POP 2
Queue POP 3
Queue POP 4
Queue POP 5
```

Standard Template Library (STL) in C++ - Set

- Store elements without repeating values
- Operations: add, remove, search

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    set<string> S;
    S.insert("abc"); S.insert("def"); S.insert("xyz");
    S.insert("abc");

    set<string>::iterator p;
    for(p = S.begin(); p != S.end(); p++) cout << *p << " ";
    cout << endl;
}
```



abc def xyz

Standard Template Library (STL) in C++ - Set

- Store elements without repeating values
- Operations: add, remove, search

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    set<string> S;
    S.insert("abc"); S.insert("def"); S.insert("xyz");
    string s1 = "def";
    set<string>::iterator p = S.find(s1);
    if(p == S.end())
        cout << "xau " << s1 << " does not exist" << endl;
    else
        cout << "xau " << s1 << " exists in S" << endl;
}
```



xau def exists in S

Standard Template Library (STL) in C++ - Set

- Store elements without repeating values
- Operations: add, remove, search

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    set<string> S;
    S.insert("abc"); S.insert("def"); S.insert("xyz");
    string s1 = "xyz";
    S.erase(s1);
    set<string>::iterator p;
    for(p = S.begin(); p != S.end(); p++) cout << *p << " ";
    cout << endl;
}
```

abc def



Standard Template Library (STL) in C++ - Set

- The set structure in C++ provides the function `upper_bound(k)`: returns a pointer to the smallest element that is greater than `k` in the set. If `k` is greater than or equal to the largest element, the function returns a pointer to the position after the last element of the set

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    set<int> S;
    for(int v = 1; v <= 5; v++) S.insert(2*v);
    set<int>::iterator p = S.upper_bound(3);
    cout << "upper_bound(3) = " << *p << endl;
    p = S.upper_bound(4);
    cout << "upper_bound(4) = " << *p << endl;
    p = S.upper_bound(10);
    if(p == S.end()) cout << "no upper_bound of 10" << endl;
}
```



```
upper_bound(3) = 4
upper_bound(4) = 6
no upper_bound of 10
```

Standard Template Library (STL) in C++ - Set

- The set structure in C++ provides the function `upper_bound(k)`: returns a pointer to the smallest element that is greater than `k` in the set. If `k` is greater than or equal to the largest element, the function returns a pointer to the position after the last element of the set

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    set<int> S;
    for(int v = 1; v <= 5; v++) S.insert(2*v);
    set<int>::iterator p = S.lower_bound(3);
    cout << "lower_bound(3) = " << *p << endl;
    p = S.lower_bound(4);
    cout << "lower_bound(4) = " << *p << endl;
    p = S.lower_bound(11);
    if(p == S.end()) cout << "no lower_bound of 11" << endl;
    else cout << "lower_bound of 11 = " << *p << endl;
}
```



```
lower_bound(3) = 4
lower_bound(4) = 4
no lower_bound of 11
```


Standard Template Library (STL) in C++ - Map

- A data structure that stores pairs of (key, value)
- Operation: add a pair of (key, value); query the value corresponding to a given key.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    map<string, int> M;
    M["abc"] = 1; M["def"] = 2; M["xyzt"] = 10;
    string k = "abc";
    cout << "value of key " << k << " = " << M[k] << endl;
    for(map<string,int>::iterator p = M.begin(); p != M.end(); p++)
        cout << p->first << " is mapped to value " << p->second << endl;
    string k1 = "1234";
    cout << "value of " << k1 << " = " << M[k1] << endl;
}
```



```
value of key abc = 1
xyzt is mapped to value 10
abc is mapped to value 1
def is mapped to value 2
value of 1234 = 0
```

Standard Template Library (STL) in C++ - Priority Queue

- Store elements, retrieve the element with the largest/smallest key efficiently

```
#include <bits/stdc++.h>
#define pii pair<int,int>
using namespace std;
int main() {
    priority_queue<int> pq;
    pq.push(5);    pq.push(1);    pq.push(100);    pq.push(30);
    while(!pq.empty()){
        int e = pq.top(); pq.pop();
        cout << "pq pop " << e << endl;
    }
}
```



```
pq pop 100
pq pop 30
pq pop 5
pq pop 1
```

Standard Template Library (STL) in C++ - Priority Queue

- Store elements, retrieve the element with the largest/smallest key efficiently

```
#include <bits/stdc++.h>
#define pii pair<int,int>
using namespace std;
int main() {
    priority_queue<pii> PQ;
    PQ.push(make_pair(4,-40));
    PQ.push(make_pair(1,-10));
    PQ.push(make_pair(9,-900));
    while(!PQ.empty()){
        pii e = PQ.top(); PQ.pop();
        cout << "PQ pop (" << e.first << "," << e.second << ")" << endl;
    }
}
```



```
PQ pop (9,-900)
PQ pop (4,-40)
PQ pop (1,-10)
```

Standard Template Library (STL) in C++ - Priority Queue

- Store elements, retrieve the element with the largest/smallest key efficiently

```
#include <bits/stdc++.h>
#define pii pair<int,int>
using namespace std;
int main() {
    priority_queue<pii, vector<pii>, greater<pii> > PQ;
    PQ.push(make_pair(4,-40));
    PQ.push(make_pair(1,-10));
    PQ.push(make_pair(9,-900));
    while(!PQ.empty()){
        pii e = PQ.top(); PQ.pop();
        cout << "PQ pop (" << e.first << "," << e.second << ")" << endl;
    }
}
```



```
PQ pop (1,-10)
PQ pop (4,-40)
PQ pop (9,-900)
```

A large graphic on the left side of the slide. It features a dark blue background with a circular pattern of red dots of varying sizes, creating a sense of depth and movement. The word "HUST" is centered within this pattern in a bold, white, sans-serif font.

HUST

THANK YOU !