ĐẠI HỌC
BÁCH KHOA

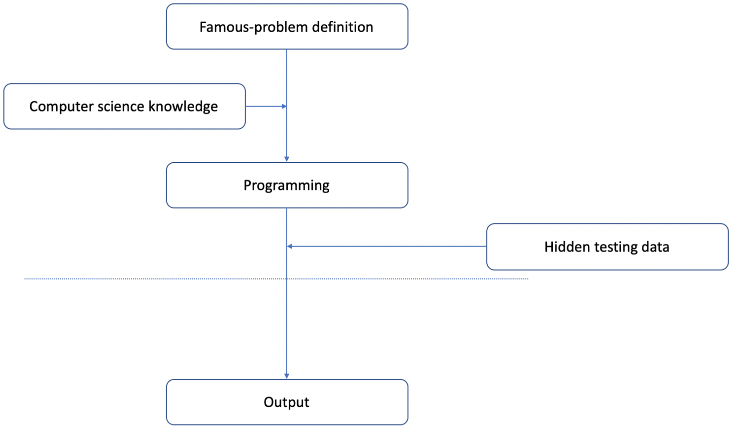
25 YEARS ANNIVERSARY
SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

APPLIED ALGORITHMS

Algo and DST Thinking
+ Implementation Skills

Introduction to programming exercises



Main objective: Proposed correct algorithms and executable programs fastest as much as possible!

Objectives

- Given a real-life programming problem, the students can
  - propose problem formulations,
  - propose the right efficient algorithms and data structures for the problem; and then
  - translate the proposed algorithms and data structures to source code and executable programs;

  in a predefined time amount such that the program returns the correct solutions in limited computation and memory resources.
- The objective of this course is to provide and to enhance these skills (the data structure, algorithm and programming skills to solve real-life problems).

Approaches

- Study common classes of programming problems
- Find out applications of data structures and algorithms based on knowledge in
  - the fundamental algorithm course,
  - the fundamental data structure course
- Study and present other common classes of algorithms and data structures
- Study simple maths and informatics theory related to solving problem by computer
- Practice algorithm and data-structure proposal skills
- Enhance the programming skill
- Practice more and more these skills

References

- Competitive Programming. Steven Halim `http://libgen.io/ads.php?md5=f6f195012783a8b3c8bb7628882a51b7`
- Slides bài giảng Phân tích và thiết kế thuật toán. Nguyễn Đức Nghĩa
- Algorithm design. Jon Kleinberg and Éva Tardos.
- Introduction to Algorithms. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein.
- Bài giảng Chuyên đề Lê Minh Hoàng
- Competitive Programming Course at Reykjavík University

Contents

- Tentative plan

No.	Timeline	Topics	Activities
1		Introduction	
2		Data structure and libraries	
3		Lab	
4		Recursion and Branch-and-bound algorithm	
5		Conquer and divide	
6		Dynamic programming	
7		Lab	
8		Lab and mid-term test	
9		Dynamic programming	
10		Graph	
11		Lab	
12		Graph	
13		String processing	
14		Lab	
15		NP-hard problems	

Exercise templates

- In most exercises, an exercise contains:
  - Problem description
  - Input description
  - Output description
  - Examples of input and output
  - Computation time limit in seconds
  - Memory limit in bytes/megabytes
- Task is to write an executable program to solve problem instances (different inputs) as many as possible.
  - The input data, in default, is correct; the program is not required to check its correctness.
  - The program runs in the given computation time and memory limits.

Example of a problem – ALICEADD

Problem description

Alice has a candies, Bob gives Alice more b candies. How many candies does Alice have in total?

Input description

The first line contains an integer T $(1 \leq T \leq 10)$ which is number of tests (instances). In T following lines, each of them includes a test. One test consists of 2 non-negative integers a, b, in which $0 \leq a, b \leq 10^{18}$, and they are separated by at least one space.

Output description

Outputs are written in T lines corresponding to T tests, each line is result of $a + b$ for each test.

Example of a problem – ALICEADD

Input	Output
2	8
3 5	5
4 1	

Solution to the problem

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        int a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

Solution to the problem

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        int a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is the above answer correct for all tests?

Solution to the problem

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        int a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is the above answer correct for all tests?
- What happens when $a = b = 10^{18}$?

Solution to the problem

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        int a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is the above answer correct for all tests?
- What happens when $a = b = 10^{18}$? The result value is bigger than the maximum value of an integer variable.

Solution to the problem

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        int a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is the above answer correct for all tests? NO!
- What happens when $a = b = 10^{18}$? The result value is bigger than the maximum value of an integer variable.

Solution to the problem

- When $a = b = 10^{18}$, the correct result is 2×10^{18}

Solution to the problem

- When $a = b = 10^{18}$, the correct result is 2×10^{18}
- The result value is too big for a 32-bit integer variable, so it leads to overflow.

Solution to the problem

- When $a = b = 10^{18}$, the correct result is 2×10^{18}
- The result value is too big for a 32-bit integer variable, so it leads to overflow.
- Use of 64-bit integer variables will lead to a correct solution.

Correct solution

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        long long a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

Correct solution

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        long long a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is this answer correct?

Correct solution

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        long long a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is this answer correct? RIGHT!

Correct solution

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        long long a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is this answer correct? RIGHT!
- What is solution in case the values of a and b are larger?

Correct solution

```
#include <iostream>
using namespace std;
int main() {
    int T;
    cin >> T;
    for (int i = 0; i < T; i++) {
        long long a, b;
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- Is this answer correct? RIGHT!
- What is solution in case the values of *a* and *b* are larger? Large-number processing!

Instant and online evaluation system

- codeforces.com
- vn.spoj.com
- https://ru.kattis.com/
- ...

- The system responses immediately when an answer is submitted
- Accepted programming languages:
  - C/C++
  - Java
  - Python
  - Pascal
  - ...

Instant and online evaluation system

- codeforces.com
- vn.spoj.com
- https://ru.kattis.com/
- ...

- The system responses immediately when an answer is submitted
- Accepted programming languages:
  - C/C++
  - Java
  - Python
  - Pascal
  - ...

Our university uses *private server codeforces* to help students practice solving problems online and using the cms system combined with the automatic code matching system to check plagiarism in the exams.

Responses from the instant and online evaluation systems

- The responses are simple, as follows:
  - Accepted
  - Wrong Answer
  - Compile Error
  - Run Time Error
  - Time Limit Exceeded
  - Memory Limit Exceeded
- Some servers give a response for each test, but in general, they just return simple responses as above.

A. Problem-reading skill

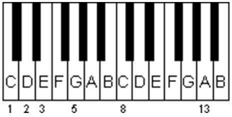
- Problem background: This represents applications of the problem in the real life
- Problem description
- Input and output description
- Examples of input and output for simple test.
- Computation time and memory limits for tests.

Is the problem background important?

- NO?
  - Example: *"Write a program to sort n real numbers. Memory limit is 3MB. Computation time limit is 1 second. Limit of input size is $n \leq 10^6$. Limit of real number value is in 4 bytes real number range with input format with exactly two digits after floating point.*
  - Short and clear!
- Informatics: MUST HAVE!!!

Is the problem background important?

- Informatics: MUST HAVE!!!
  - Ví dụ: *The Fibonacci sheet music is a piece of music whose melody is derived from one of the most famous sequence of numbers in Number Theory - the Fibonacci numbers. The first two numbers in the sequence are 1 and 2, and the subsequent numbers are determined by the sum of the two consecutive numbers immediately preceding it in the sequence. The Fibonacci sheet music is obtained by converting the Fibonacci sequence into a sequence of notes following the rule of converting a positive integer into a musical note later:*

    Click here
  - Raise the motivation for students (problem solver)
  - An application of a computer science topic

B. Problem-classification skill

- Classify quickly problems into popular distinct classes
- Combine different classes of problems

- Example of popular classes:

Classes	Sub-classes
Ad-Hoc	Direct, Simulation
Exhaustive search	Loop, Backtracking, Branch-and-bound
Conquer and divide	Traditional and new approaches
Conquer and decrease	Traditional and new approaches
Greedy	Traditional and improved approaches
Dynamic programming	Traditional and improved approaches
Graph	Traditional and improved approaches
Special graph	Traditional and improved approaches
Maths	Traditional and improved approaches
String processing	Traditional and improved approaches
Geographic computation	Traditional and improved approaches
Some special algorithm groups	

C. Algorithm-analysis skill

- The solution should be fast enough and not use too much memory.
- The solution should be as simple as possible.
- Use algorithm analysis method to determine if the expected solution satisfies memory and computation time limit.
- In general, a simple personal computer can do 5×10^8 calculations per second.
- For example, sorting $n \leq 10^6$ integer numbers in 3 seconds
  - Can the bubble sort algorithm ($O(n^2)$) solve completely that task?
  - Can a quick sort and complicated algorithm ($O(n \log n)$) solve completely that task?
- For example, sorting $n \leq 10^3$ integer numbers in 1 seconds
  - So, can the bubble sort algorithm ($O(n^2)$) solve completely that task?
- Remember that the simplest algorithm satisfying all requirements in computation time and memory resources is the best choice.

- Practice the computation time approximation
- Hint:
  - $2^{10} \approx 10^3$
  - $log_2 10 \approx 3$
- In the case, you are not sure about the correctness of the algorithm, please find a proof for that algorithm. Even if you can not prove or disprove it, you still can understand more the problem.

n	Complexity	Example
≤ 10	$O(n!)$, $O(n^6)$	Enumerate all permutations
≤ 15	$O(2^n \times n^2)$	Dynamic programming for TSP
≤ 20	$O(2^n)$, $O(n^5)$	Enumerate binary permutations, Dynamic programming
≤ 50	$O(n^4)$	3&4-dimension dynamic programming, Enumerate combinations ${}_nC_k = 4$
$\leq 10^2$	$O(n^3)$	Floyd Warshall
$\leq 10^3$	$O(n^2)$	Bubble/Insertion/Selection sort
$\leq 10^5$	$O(n \log_2 n)$	Merge sort, Segment/Interval tree
$\leq 10^6$	$O(n)$, $O(\log_2 n)$, $O(1)$	Input reading $n \leq 10^6$

D. Skill to master programming languages

- Master popular programming languages (at least one)
- Master built-in libraries
  - C++'s Standard Template Library
  - The Java Class Library
- If it is already in the standard library, you don't need to re-implement it, you should use libraries to do program faster and bug free.
- Limitations: the library's customizability is not high. A lot of the algorithmic processing parts cannot use the library directly, but they must be customized.

E. Variable naming skill

- Names should be meaningful
- Function name starts with capital letter, for example, `void ReadInp()`
- All letter in a constant name are capital, for example, const int MAX=100000;
- Array name starts with a letter `a`, following by a capital letter, for example, `int aCost[MAX];`
- String-variable names start with a letter `s`, following by a capital letter, for example, `string sLine[100];`
- Name of a `vector` variable starts a letter `v`, following by a capital letter, for example, `vector<int> vList;`
- Name of a single variable name should contain lower letters, for example, `int i, u, sum, res;`
- Variable names should be as close to the names/nous given in the problem as possible.
- ...

F. Skill to test the program

- Test to make sure the outputs of the program are correct and the program runs in the limited time amount.
- Try to diagnose the solution to the problem by finding counterexamples (an input where the solution returns incorrect, or takes too long time to find the result).
- Do experiments with boundary-value instances and large-size instances.
- Write a simple program to check the correctness of solutions for small-size tests.

G. Typing skill

- Become a faster and more accurate keyboard typist;
- Don't let keyboard typing limit your problem solving;
- During typing without looking at the keyboard, eyes looking at the screen for checking the correctness of typing, while the head can still think in parallel to the next problem.

- TypeRacer is an efficient and interesting tool for practicing typing: http://play.typeracer.com/

H: Practice, practice more and practice forever

- The more you practice, the more you improve your problem solving and programming skills.
- There are many websites to help you practice solving problems from past tests.
- There exist some websites which regularly host competitions such as UVa, Codeforces, TopCoder, Kattis.

Programmers participating regularly competitions, like athletes, need regular practice to stay inspired and develop problem solving skills!

Ad Hoc Problems

Adhoc problems

- A class of simple and easy problems
- Solution can be derived directly problem descriptions
- Direct or simulation
- Computation time limit does not matter
- Sometime the description is long, unclear and confusing
- Some complex problems cause difficulties in programming

Problem: Cut down store

The COVID19 epidemic made people very limited out of their homes, which pushed many companies out of business and many others had to cut down their stores, offices, and to fire employee, to reduce salary and bonus budget.

XTEC Company is not an exception. It has 4 stores and decides to cut down 2 of them with the lowest negative profits in 2019.
 bf Requirements: Given the 2019 profit of the 4 stores, please indicate the total negative profit of the stores that must be closed.

Cut down store

Input

- The first line contains an integer T ($T < 20$), which is the number of tests.
- In T following lines, each of them contains 4 distinct integers representing profits of the stores in 2019. The values of the integers are in $[-10000, 10000]$.

Kết quả ra

The result of each test is written in a line, which is the total negative profit of the stores that must be closed.

Problem: Cut down store

Input	Output

3
-1000 2000 3000 -4000
3000 -2500 1500 100
-1500 -1200 -1800 -1900
	-5000
	-2500
	-2700

Solution to the problem "Cut down store"

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);

    int aProfit[4];
    int T; cin >> T;
    for (int i = 0; i < T; i++) {
        cin >> aProfit[1] >> aProfit[2];
        cin >> aProfit[3] >> aProfit[4];
        sort(aProfit, aProfit + 4);
        int sum=0;
        if (aProfit[1]<0) sum+=aProfit[1];
        if (aProfit[2]<0) sum+=aProfit[2];
        cout << sum << endl;
    }
    return 0;
}
```

Problem: SMS typing

Mobile phones have become an indispensable part of modern life. In addition to making calls, mobile phones can send messages that are called as SMS. Unlike computer keyboards, most mobile phones have limited number of keys. To be able to type all the letters of the alphabet, multiple letters are displayed on the same key. Therefore, to type certain letters, a key must be pressed repeatedly until the desired letter appears on the screen.

Given a text, you are asked for counting the number of keystrokes to display whole text on the screen.

Problem: SMS typing

The assumption, the layout of keys is as follows:

	abc	def
ghi	jkl	mno
pqrs	tuv	wxyz
	<SP>	

In the table, a cell represents one key. <SP> represents the space key. To display letter 'a', you will have to press the corresponding key 1 time, but to display 'b' of the same key, you will have to press twice continuously and for 'c' key 3 times. Similarly, press 1 time for 'd', twice for 'e' and 3 times for 'f'. The other letters are also done in the same way. Note that to create a space, you need to press the space key once.

Problem: SMS typing

Input

The first line is an integer T, which is the number of tests. In T following lines, each line contains a text with spaces and lower case letters. Each line has at least one letter and at most 100 letters.

Output

The result of a test is written in a line in the output. Each line begins with the test order and is followed by a number indicating the number of keystrokes that produce the corresponding text. Please See the example output for the correct format!

Problem: SMS typing

Input	Output

`2`
`welcome to ulab`
`good luck and have fun`
	`Case #1: 29`
	`Case #2: 41`

Solution to the problem “SMS typing”

```
#include <cstdio>
#include <iostream>
#include <string>
using namespace std;
string keys[12] = {
    "",      "abc", "def",
    "ghi",   "jkl", "mno",
    "pqrs",  "tuv", "wxyz",
    "",      " ",   ""
};
int main() {
    int T;
    scanf("%d\n", &T);
    for (int t = 0; t < T; t++) {
        // Each test case is handled here
    }
    return 0;
}
```

Solution to the problem “SMS typing”

```
    // Each test case:
    string line;
    getline(cin, line);
    int cnt = 0;
    for (int i = 0; i < line.size(); i++) {
        int cur;
        for (int j = 0; j < 12; j++) {
            for (int k = 0; k < keys[j].size(); k++)
                if (line[i] == keys[j][k]) {
                    cur = k + 1;
                }
            }
        }
        cnt += cur;
    }
    printf("Case #%d: %d\n", t + 1, cnt);
```

Practical exercises

- ALICEADD
- SUBSEQMAX

SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!