



CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

GIỚI THIỆU MÔN HỌC



**Data Structure and
Algorithm**

Giảng viên: Th.S Bùi Văn Kiên



TỔNG QUAN MÔN HỌC

- Thông tin môn học

Tên MH: Cấu trúc dữ liệu & giải thuật

Số TC: 3.

- Thông tin giảng viên

Th.S Bùi Văn Kiên - Bộ môn CNPM, Khoa CNTT1.

Email: kienbv@ptit.edu.vn



TỔNG QUAN MÔN HỌC

■ Tỷ lệ đánh giá

- Chuyên cần: 10%

Nghỉ 1 buổi không có lý do trừ 1.5 điểm.

- Kiểm tra 1: 10% (Thi máy)
- Kiểm tra 2: 10% (Thi giấy)
- Kiểm tra 3: 10% (Thi máy)

Điểm cộng: phát biểu trên lớp + thực hành (mỗi lần 1 điểm)

Điểm trừ: gian lận thực hành /thi cử

- Thi cuối kì (trên máy): 60%



Tài liệu tham khảo và Công cụ hỗ trợ

■ TÀI LIỆU THAM KHẢO

- [1] Bài giảng Cấu trúc dữ liệu và Giải thuật, Nguyễn Duy Phương, Nguyễn Mạnh Sơn, 2020
- [2] Giải thuật và Lập trình, Lê Minh Hoàng
- [3] Competitive programming 3

■ CÔNG CỤ HỖ TRỢ

- Cổng thực hành trực tuyến <http://code.ptit.edu.vn>
- Tài khoản:
 - ❖ Username: Mã sinh viên
 - ❖ Pass mặc định: Ngày – tháng – năm sinh (ddmmyyyy)



HỌC TẬP NHƯ THẾ NÀO

- Quy chế lớp học:
 - Đi muộn 15 phút, chờ hết tiết mới được vào lớp.
 - Không làm việc riêng, không gây mất trật tựNếu vi phạm → được mời ra khỏi lớp.
 - Mang laptop + giấy nháp khi đi học (kiểm tra giấy nhiều)
 - Gọi lên phát biểu, không trả lời được → bị trừ điểm kiểm tra
 - Gian lận khi kiểm tra → 0 điểm
- Xin nghỉ có phép:
 - nhắn trực tiếp mã sinh viên + lí do lên nhóm Zalo của lớp

HỌC TẬP NHƯ THẾ NÀO

- Điều kiện miễn môn học (10 điểm):

Tham gia luyện tập contest trên Codeforces.com, rank ≥ 1700



HOME

TOP

CATALOG

CONTESTS

GYM

PROBLEMSET

GROUPS


RATING

EDU

API

CALENDAR

HELP

RAYAN 

RATING

RATING (ALL)

FRIENDS RATING

Country:

any country












City:

any city

Organization:

PTIT Vietnam, 96

Rating

	Who	#	=
1 (3530)	 AkiLotus	145	2017
2 (3696)	 DuongForeverAlone	118	2002
3 (6460)	 VanLam	68	1826
4 (8465)	 _Alice_in_Borderland_	27	1735
5 (14676)	 nguyenthaiduong	64	1588
6 (17186)	 _phamductuan_	9	1532
7 (18319)	 nktan	6	1510
8 (19884)	 vanquyvct	25	1484
9 (22690)	 _DS	26	1444
10 (23830)	 __dribbler	15	1430
11 (25030)	 ITIS_Code	8	1416



NỘI DUNG

1

CHƯƠNG 1: ĐỘ PHỨC TẠP THUẬT TOÁN

2

CHƯƠNG 2: SẮP XẾP VÀ TÌM KIẾM

3

CHƯƠNG 3: THAM LAM & QUY HOẠCH ĐỘNG

4

CHƯƠNG 4: NGĂN XẾP, HÀNG ĐỢI

5

CHƯƠNG 5: CÂY NHỊ PHÂN

6

CHƯƠNG 6: ĐỒ THỊ



CÁC TRANG LUYỆN TẬP THÊM

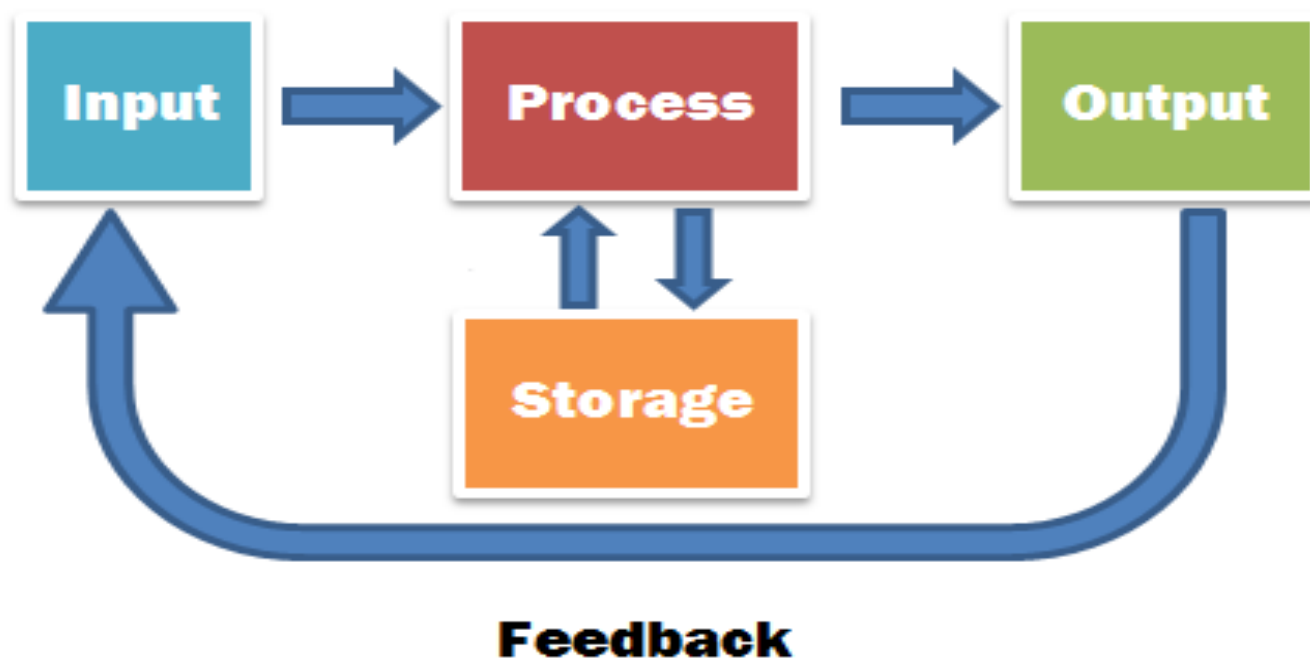
- <https://www.spoj.com/PTIT/>
- [**https://codeforces.com/**](https://codeforces.com/)
- <https://vnoi.info/>
- <https://www.hackerrank.com/>
- <https://www.geeksforgeeks.org/>
- <https://codelearn.io/>
- [**https://leetcode.com/**](https://leetcode.com/)
- [**atcoder.jp**](https://atcoder.jp)



Học viện
Công nghệ Bưu chính Viễn thông

LUYỆN TẬP LẬP TRÌNH TRỰC TUYẾN

Ví dụ về một chương trình đơn giản





Chấm tự động

- Chạy chương trình với input cho trước, so sánh với output chuẩn.
- Hệ thống trả về **AC**, nghĩa là chương trình của bạn đã pass hết tất cả các test case.
- Nên coi kết quả lỗi trả về của phần mềm chấm tự động là gợi ý để chỉnh sửa bài làm của bạn.



Chấm tự động

Các lỗi trả về:

- **No, Compilation Error (CE):** Lỗi biên dịch.
- **No, Wrong Answer (WA):** Sai kết quả. Có thể ghi ra không đúng định dạng, hoặc thuật toán (cách làm) sai.
- **No, Runtime Error (RTE):** Lỗi ngoại lệ trong thời gian chạy. Có thể do tràn mảng, chia cho 0, truy xuất vùng nhớ không hợp lệ ...
- **No, TimeLimit Exceeded (TLE):** Vượt quá thời gian chạy (tính toán độ phức tạp thuật toán chưa hiệu quả).
- **No, Memory Limit Exceeded (MLE):** Vượt quá giới hạn bộ nhớ cho phép. Có thể do đệ quy quá sâu, hoặc khai báo mảng/vector quá lớn.
- **No, Invalid Return (IR):** Chương trình trả về giá trị không mong muốn. Có thể do hàm nào đó bị dừng giữa chừng (tương tự như RTE).



Cấu trúc chương trình

Gồm 3 phần chính

- Đọc input
- Xử lý
- Ghi ra output

*** Lưu ý:**

- Cần định dạng đúng cho input và output
- Phần xử lý: tạo thói quen cho cách tư duy logic



Cấu trúc chương trình

Đọc dữ liệu đúng

- Kiểu dữ liệu phải phù hợp
- Nhớ chắc các hàm đọc dữ liệu (int, long long, string)
- Nhớ chắc cách đọc khi đề bài cho “**số bộ test**” và khi không cho số bộ test



Cấu trúc chương trình

Ghi dữ liệu đúng

- Chỉ ghi ra những gì được yêu cầu
- Chú ý khi nào cần xuống dòng
- Chú ý định dạng số, làm tròn số ... theo yêu cầu



Cấu trúc chương trình

Xử lý - Process

- Đọc kỹ giới hạn dữ liệu và mô tả đề bài;
 - Phân tích độ phức tạp để biết nên chọn thuật toán nào
 - Đôi khi cần biết giải toán trước khi code
- Code đúng thuật toán?
- Code đúng chính tả?



Học viện
Công nghệ Bưu chính Viễn thông

VÍ DỤ ĐƠN GIẢN VỀ THUẬT TOÁN



Ví dụ: Dãy số Fibonacci

Dãy số Fibonacci được định nghĩa như sau:

$$F[0] = 0$$

$$F[1] = 1$$

$$F[n] = F[n-1] + F[n-2]$$

0, 1, 1, 2, 3, 5, 8, 13, ...

Ví dụ: Dãy số Fibonacci

Dãy số Fibonacci được định nghĩa như sau:

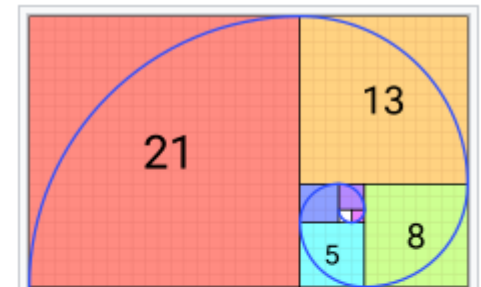
$$F[0] = 0$$

$$F[1] = 1$$

$$F[n] = F[n-1] + F[n-2]$$

Công thức tổng quát:

$$F_n = \frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^n.$$



The Fibonacci spiral: an approximation of the [golden spiral](#) created by drawing [circular arcs](#) connecting the opposite corners of squares in the Fibonacci tiling (see preceding image)



Ví dụ: Dãy số Fibonacci

Một số phần tử:

- $F[20] = 6765$
- $F[50] = 12586269025$
- $F[100] = 354224848179261915075$
- $F[500] = 1394232245616978801397243828704072839$
5007025658769730726410896294832557162286329069
1557658876222521294125



Ví dụ: Dãy số Fibonacci

Yêu cầu:

- Cho số nguyên không âm n
- Dữ liệu vào: Số n ($0 \leq n \leq 92$)
- Dữ liệu ra: Số Fibonacci thứ n
- Thời gian yêu cầu: 1s.



Ví dụ: Dãy số Fibonacci

Giải thuật 1 (navie – ngây thơ):

```
#include <iostream>
using namespace std;

long long fibo(int n) {
    if(n < 2) {
        return n;
    }
    return fibo(n-1) + fibo(n-2);
}

int main() {
    cout << fibo(10) << endl;
    return 0;
}
```



Ví dụ: Dãy số Fibonacci 1

Đánh giá giải thuật 1:

- Ngắn gọn.
- Dễ triển khai, tư duy theo đúng yêu cầu và mô tả của bài toán.
- Kết quả phản hồi trong dưới 1s với $n \leq 40$. Tuy nhiên, với n lớn, chương trình không có khả năng phản hồi ngay.



Ví dụ: Dãy số Fibonacci 1

Ước tính thời gian chạy:

- Gọi $T(n)$ là độ phức tạp thuật toán khi xây dựng số Fibonacci bằng cách dung đệ quy (không có nhớ)
- Xét $n < 2$, $T = O(1)$

```
long long fibo(int n) {  
    if(n < 2) {  
        return n;  
    }  
    return fibo(n-1) + fibo(n-2);  
}
```

$$T(n) = O(1) + T(n-1) + T(n-2)$$



Ví dụ: Dãy số Fibonacci 1

Ước tính thời gian chạy:

- Gọi $T(n)$ là độ phức tạp thuật toán khi xây dựng số Fibonacci bằng cách dung đệ quy (không có nhớ)
- Xét $n < 2$, $T = O(1)$

```
long long fibo(int n) {  
    if(n < 2) {  
        return n;  
    }  
    return fibo(n-1) + fibo(n-2);  
}
```

$$T(n) = O(1) + T(n-1) + T(n-2)$$

$$T(n) > F(n) = O(1.6180^n)$$

$$T(100) \sim 8 \cdot 10^{20}$$



Ví dụ: Dãy số Fibonacci 1

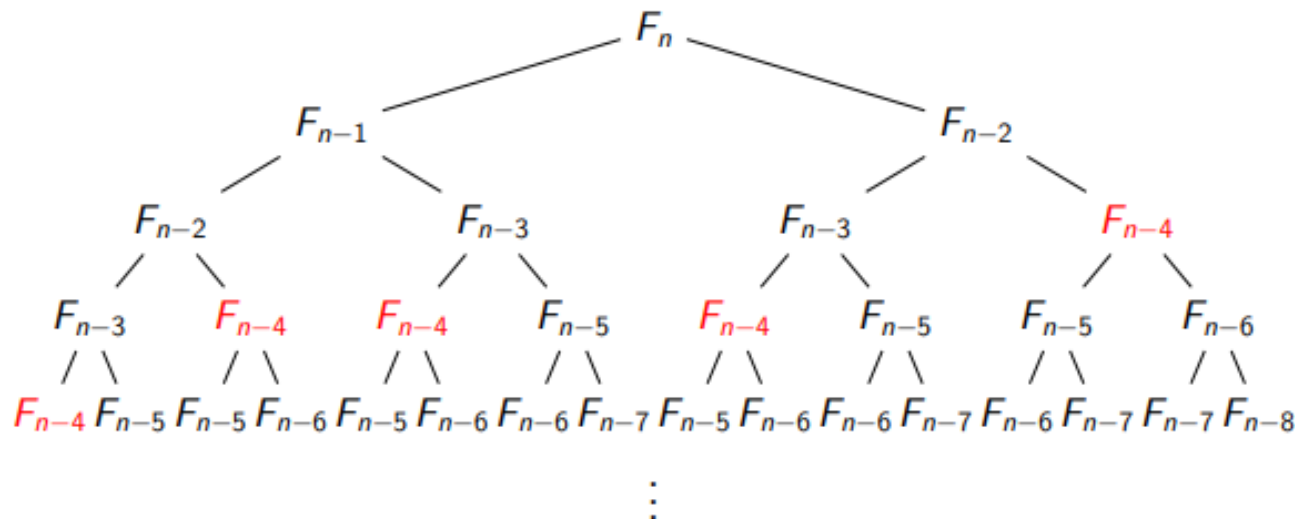
Đánh giá giải thuật 1:

- Ngắn gọn.
- Dễ triển khai, tư duy theo đúng yêu cầu và mô tả của bài toán.
- Kết quả phản hồi dưới 1s với $n \leq 40$. Tuy nhiên, với n lớn, chương trình không đưa ra đáp án ngay được (TLE).
- Xét $N = 100$, máy tính có tốc độ (1Ghz) có khả năng thực hiện 1 tỷ phép tính / giây. Theo ước tính sẽ mất 56.000 năm để chương trình đưa ra kết quả
=> mất quá nhiều thời gian, cần thuật toán tốt hơn để xử lý.

Ví dụ: Dãy số Fibonacci 1

Lý do thuật toán 1 chạy chậm:

- Số lượng lời gọi hàm quá lớn và trùng lặp
- Để tính $F(n)$, cần phải tính $F(n-1)$ và $F(n-2)$. Để tính $F(n-1)$ cần tính $F(n-2)$ và $F(n-3)$, ... \Rightarrow cây đệ quy





Ví dụ: Dãy số Fibonacci 2

Giải thuật 2:

- Tính toán bằng cách liệt kê từng phần tử của dãy số
0, 1, 1, 2, 3, ...
- Phần tử tiếp theo là $1 + 2 = 3$



Ví dụ: Dãy số Fibonacci 2

Giải thuật 2:

- Tính toán bằng cách liệt kê từng phần tử của dãy số
0, 1, 1, 2, 3, ...
- Phần tử tiếp theo là $1 + 2 = 3$

```
long long fibo_array(int n) {  
    vector<long long> f;  
    f.push_back(0);  
    f.push_back(1);  
    for (int i = 2; i <= n; i++) {  
        long long next_number = f[i-1] + f[i-2];  
        f.push_back(next_number);  
    }  
    return f[n];  
}
```



Ví dụ: Dãy số Fibonacci 2

Giải thuật 2:

- Độ phức tạp: $O(n)$

```
long long fibo_array(int n) {  
    vector<long long> f;  
    f.push_back(0);  
    f.push_back(1);  
    for (int i = 2; i <= n; i++) {  
        long long next_number = f[i-1] + f[i-2];  
        f.push_back(next_number);  
    }  
    return f[n];  
}
```

Ví dụ: Dãy số Fibonacci 2

Giải thuật 3: Đệ quy có nhớ

- Độ phức tạp: $O(n)$
- Số lần gọi đệ quy: $2n-1$

```
long long fibo(int n) {  
    if(n < 2) {  
        return n;  
    }  
    return fibo(n-1) + fibo(n-2);  
}
```

```
long long F[105];  
  
long long fibo(int n) {  
    if(F[n] != 0) return F[n];  
    if(n < 2) return n;  
    long long temp = fibo(n-1) + fibo(n-2);  
    F[n] = temp;  
    return F[n];  
}
```



Ví dụ: Dãy số Fibonacci 3

Làm thế nào để tính được số Fibonacci lớn ($F[500] \dots$)

- Vấn đề tràn số

→ Xử lí số nguyên lớn

(xây dựng class số nguyên lớn với `char[]`, `string`, `array`)



Ví dụ: Dãy số Fibonacci 4

Làm thế nào để tính được số Fibonacci lớn hơn nữa?

- Tính số Fibonacci thứ 10^{12} . Đáp án có thể rất lớn, chỉ cần kiểm tra theo module 1.000.000.009

→ Thuật toán $O(n)$ không đáp ứng được thời gian

- Vấn đề thời gian chạy?
- Cách thức lưu trữ kết quả
- Làm thế nào để “chấm tự động”?



Ví dụ 2: Sàng nguyên tố

Đề bài

Problem A: Số siêu nguyên tố

Timelimit: 1s

Số P được gọi là số siêu nguyên tố, nếu nó là số nguyên tố và khi ta lần lượt loại bỏ các chữ số hàng đơn vị (từ phải sang trái) thì số mới nhận được vẫn là một số nguyên tố.

Ví dụ:

23 là số siêu nguyên tố vì 23 và 2 là số nguyên tố.

431 không phải là số siêu nguyên tố vì 431, 43 là số nguyên tố, nhưng 4 thì không là số nguyên tố.

Input:

Cho 2 số nguyên L và R.

Output:

In ra một số nguyên là số lượng số siêu nguyên tố trong đoạn [L, R].

Giới hạn:

Subtask 1 (50%): $1 \leq L, R \leq 10^7$,

Subtask 2 (50%): $1 \leq L, R \leq 10^9$.

Test ví dụ:

Input	Output
1 7	4



Ví dụ 2: Sàng nguyên tố

Kiểm tra số nguyên tố thông thường:

```
bool isPrime(int n) {  
    for (int i = 2; i <= sqrt(n); i++)  
        if (n % i == 0) {  
            return false;  
        }  
    return n > 1;  
}
```

Ví dụ 2: Sàng nguyên tố

Kiểm tra số nguyên tố sử dụng sàng

```
void sieve(int N) {
    bool isPrime[N+1];
    for(int i = 0; i <= N; ++i) {
        isPrime[i] = true;
    }
    isPrime[0] = false;
    isPrime[1] = false;
    for(int i = 2; i * i <= N; ++i) {
        if(isPrime[i] == true) {
            // Mark all the multiples of i as composite
            for(int j = i * i; j <= N; j += i)
                isPrime[j] = false;
        }
    }
}
```

	2	3	4	5	6	7	8	9	10	Prime numbers
11	12	13	14	15	16	17	18	19	20	
21	22	23	24	25	26	27	28	29	30	
31	32	33	34	35	36	37	38	39	40	
41	42	43	44	45	46	47	48	49	50	
51	52	53	54	55	56	57	58	59	60	
61	62	63	64	65	66	67	68	69	70	
71	72	73	74	75	76	77	78	79	80	
81	82	83	84	85	86	87	88	89	90	
91	92	93	94	95	96	97	98	99	100	
101	102	103	104	105	106	107	108	109	110	
111	112	113	114	115	116	117	118	119	120	

Primes[] = true

i = 2, isPrime[2] = true

→ For (j = 2*2 = 4 → N) isPrime[j] = false; → 4, 6, 8, 10, **12**, ... là hợp số

i = 3, isPrime[3] = true

→ For (j = 3*3 = 9 → N) isPrime[j] = false; → 9, **12**, 15, 18, ... là hợp số

Ví dụ 2: Sàng nguyên tố

Độ phức tạp của thuật toán:

Số lần lặp của vòng lặp trong là:

Khi $i=2$, vòng lặp trong lặp $N/2$ lần.

Khi $i=3$, vòng lặp trong lặp $N/3$ lần.

Khi $i=5$, vòng lặp trong lặp $N/5$ lần.

...

Độ phức tạp tổng:

$N \cdot (1/2 + 1/3 + 1/5 + \dots) = O(N \log N)$.

	2	3	4	5	6	7	8	9	10	Prime numbers
11	12	13	14	15	16	17	18	19	20	
21	22	23	24	25	26	27	28	29	30	
31	32	33	34	35	36	37	38	39	40	
41	42	43	44	45	46	47	48	49	50	
51	52	53	54	55	56	57	58	59	60	
61	62	63	64	65	66	67	68	69	70	
71	72	73	74	75	76	77	78	79	80	
81	82	83	84	85	86	87	88	89	90	
91	92	93	94	95	96	97	98	99	100	
101	102	103	104	105	106	107	108	109	110	
111	112	113	114	115	116	117	118	119	120	

Kiểm tra từ $1 \rightarrow N$ có các số nào là số nguyên tố?

Cách làm cũ: $\sqrt{N} \cdot N$

$N = 10^6 \rightarrow$ tương đương, $N = 10^7$

\rightarrow có khác biệt lớn khi dùng sàng nguyên tố



Bài học từ ví dụ

- Nếu không có kiến thức về thuật toán mà chỉ “nghĩ sao code vậy” một cách “ngây thơ” thì sẽ rất dễ rơi vào tình huống làm bài toán WA, TLE, MLE, ...

→ Kết quả thu được từ môn học:

- Các mô hình thuật toán:
 - Các mô hình thuật toán
 - Cách tiếp cận để giải quyết bài toán
 - Cách phân tích đánh giá để lựa chọn giải pháp phù hợp
 - Tư duy logic, rõ ràng, nhanh gọn.
- Các cấu trúc dữ liệu:
 - Phương pháp tổ chức lưu trữ dữ liệu phù hợp với yêu cầu
 - Các bài toán đặc trưng và hướng giải quyết phù hợp
 - Tư duy mở: cùng một bài toán nhưng với các kích thước dữ liệu khác nhau → cần sử dụng cấu trúc dữ liệu và thuật toán hoàn toàn riêng biệt.



QUESTIONS



THANK YOU!