Lập trình để thi đấu
Competitive programming

IOI, ACM-ICPC, Olympic Tin học sinh viên, …

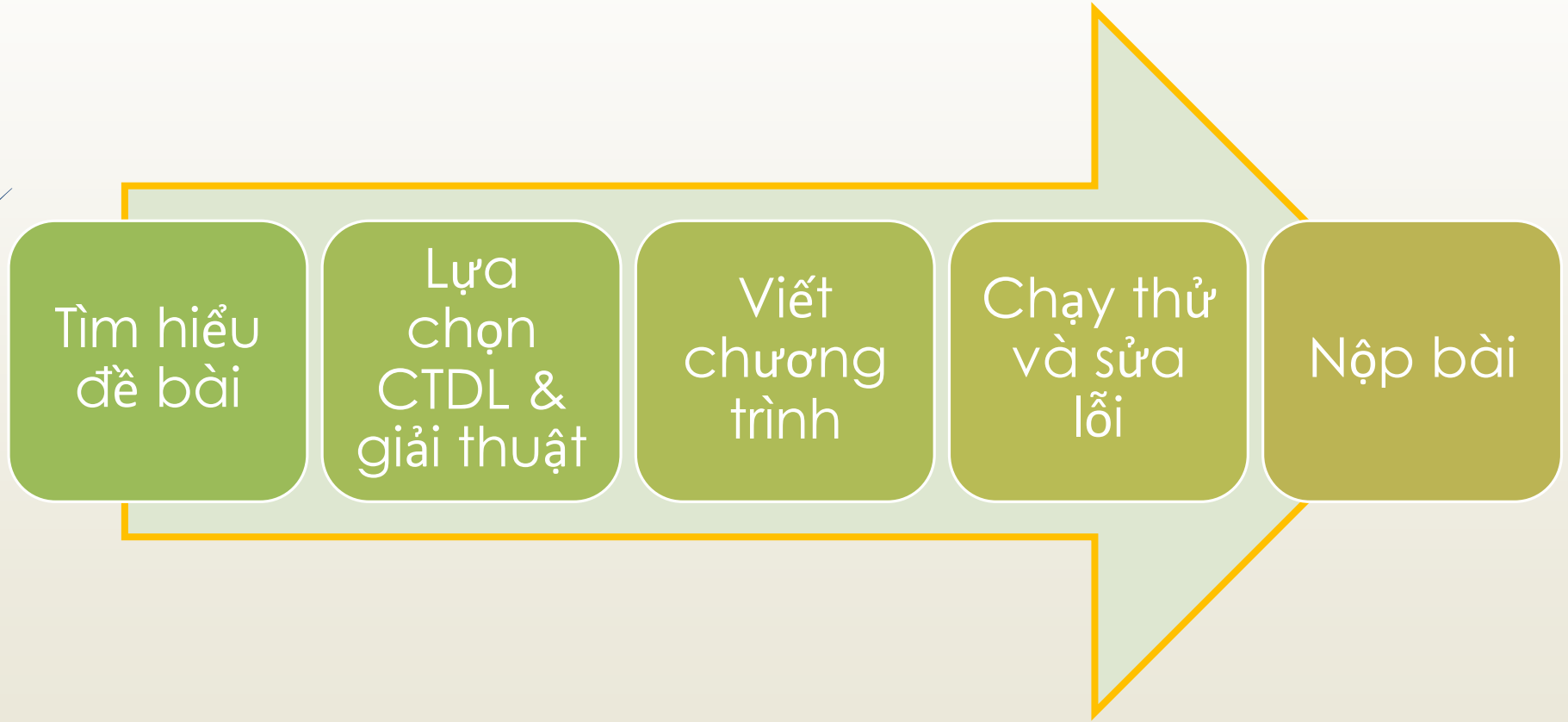
Nội dung

- Đặc trưng của phương pháp lập trình để thi đấu
- Các giải pháp cải thiện hiệu quả làm việc
  - Thời gian lập trình
  - Thời gian thực thi của chương trình

Đặc trưng

- Lập trình với các bài toán đã được có "phương pháp giải" trong thời gian càng nhanh càng tốt.
- Phương pháp giải cần có tính khả thi trong điều kiện giới hạn về thời gian và các tài nguyên khác.
- Nhanh: thời gian lập trình (programming time) vs thời gian thực thi (execution time).
- Tip: developer >> programmer >> coder

Thời gian lập trình



Cải thiện thời gian lập trình (1)

- Đọc và phân tích đề bài
  - Đọc hiểu phần mô tả
  - Xác nhận lại qua dữ liệu mẫu
- Chọn giải thuật → CTDL
  - Đoán nhận giải thuật qua ví dụ, thử nghiệm với dữ liệu nhỏ
  - Đoán nhận giải thuật qua kinh nghiệm, nhanh chóng xác định bài toán thuộc vào "Dạng (loại)" nào trong các dạng đã biết, đánh giá nhanh độ phức tạp của thuật toán.
- Tip: giải thuật thô sơ – cài đặt nhanh, hiệu quả thấp ↔ giải thuật tinh chuẩn – cài đặt khó, hiệu quả cao

Cải thiện thời gian lập trình (2)

- Viết chương trình
  - Chọn NNLT thích hợp, có thể biết nhiều NNLT nhưng ít nhất phải là "bậc thầy" trên 1 NNLT
  - Sử dụng thành thạo IDE (mức độ chuyên gia)
  - Đánh máy càng nhanh càng tốt
  - Đặt các cụm từ thường sử dụng thành các từ viết tắt
  - Sử dụng một số KTLT đặc thù
- Chạy thử và sửa lỗi
  - Tạo bộ dữ liệu kiểm tra (test) (mức độ chuyên gia)
  - Gỡ rối (debug)

Ngôn ngữ lập trình

- Chọn NNLT thích hợp hỗ trợ nhiều nhất cho giải thuật (vd: java bignum; C scanf/printf, …)
- Có thể hiểu biết nhiều NNLT nhưng cần hiểu rõ và sử dụng thành thạo ít nhất 1
  - Viết ngắn gọn và hiệu quả
  - Khai thác thư viện sẵn có càng nhiều càng tốt
  - Cập nhật các thay đổi của NNLT được cộng đồng ACM thừa nhận (vd: C++17)

Phân loại bài toán – một số dạng thường gặp trong ICPC (1)

1. Ad Hoc: các bài toán không có lời giải chuẩn mực, bài toán giả lập, bài toán không phân loại được.
2. Vét cạn: các bài toán cần quét toàn bộ không gian lời giải: đếm, sinh tổ hợp, tìm kiếm quay lui, thử và sai, …
3. Chia để trị
4. Tham lam
5. Qui hoạch động

Phân loại bài toán – một số dạng thường gặp trong ICPC (2)

6. Đồ thị
    1. Cơ bản: đếm bậc, tìm kiếm, duyệt, sắp xếp tôp, xác định miền liên thông/liên thông mạnh, xác định cầu-đỉnh khớp, cây tối đại ngắn nhất, đường đi ngắn nhất, chu trình Euler, chu trình Hamilton, tô màu…
    2. Nâng cao: luồng, cặp ghép, song liên thông, siêu đồ thị, …
7. Phương pháp toán
8. Xử lý xâu (chuỗi) ký tự
9. Hình học tính toán

Các dạng bài tập trong các kỳ thi ACM ICPC Asia gần đây

STT	Dạng bài	Số bài/đề
1	Ad Hoc	0-3
2	Vét cạn	0-2
3	Chia để trị	0-1
4	Tham lam	0-1
5	Qui hoạch động	1-3
6	Đồ thị	1-2
7	Phương pháp toán	1-2
8	Xử lý xâu	1
9	Hình học tính toán	1
10	Một số bài khó	0-1

Cải thiện thời gian thực thi

- Chọn thuật giải thích hợp
  - Thành thạo nhiều giải thuật khác nhau
- Chọn CTDL thích hợp
  - Vận dụng thành thạo nhiều CTDL khác nhau trên mỗi giải thuật đã biết kèm theo đánh giá hiệu năng
- Phân tích độ phức tạp của thuật toán
  - Một số giá trị xấp xỉ: $2^{10} \approx 10^3$, $2^{20} \approx 10^6$, $2^{31} - 1 \approx 2 \times 10^9$, $2^{63} - 1 \approx 9 \times 10^{18}$
  - Qui tắc + với các lệnh nối tiếp và x với các lệnh lồng nhau
  - Độ phức tạp của 1 số bài toán quen thuộc: duyệt mảng, duyệt cây, DP, DFS, BFS, …
- Vận dụng các KTLT nâng cao

Tài liệu tham khảo

1. Robert Sedgewick – *Algorithms* – Addison Wesley 1984 (1946)
2. Steven & Felix Halim – ***Competitive programming 1*** – NUS 2010

Giải bài tập

1. Tóm tắt đề bài Combined Number:
   - Input: 2 số nguyên lớn (30 chữ số) – A, B
   - Output: 2 số nguyên lớn (60 chữ số) được trộn từ A & B – C là số bé nhất và D là số lớn nhất
2. Đoán nhận giải thuật bài Missing Bit: thử với các số K = 1, 2, 3, … sẽ đoán được giải thuật là xét sự tương quan chẵn lẻ của các số P, Q, K