Lập trình để thi đấu
Competitive programming

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

Nội dung

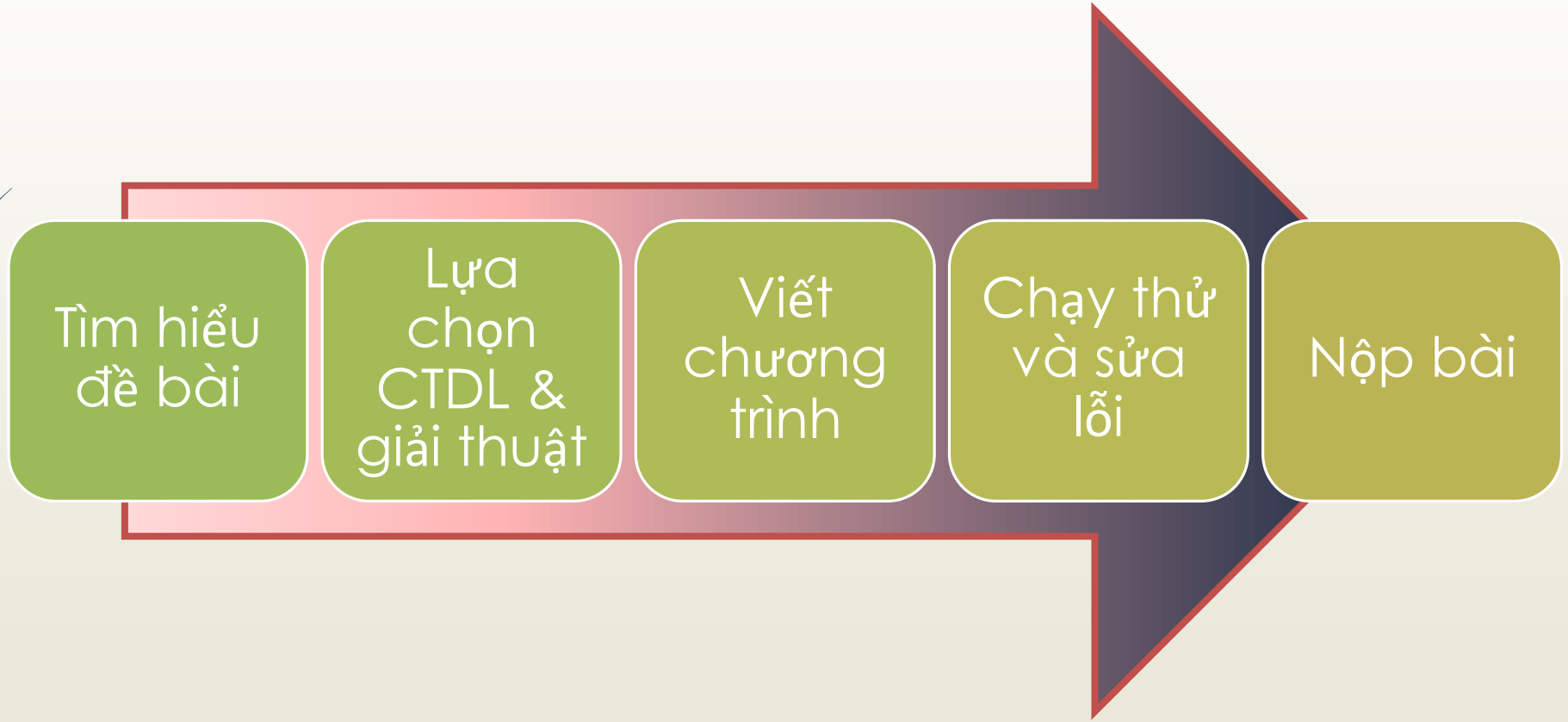
- Đặc trưng của phương pháp lập trình để thi đấu
- Các giải pháp cải thiện hiệu quả làm việc
  - Thời gian lập trình
  - Thời gian thực thi của chương trình

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.
  - Các bài làm được kiểm tra bằng các bộ test
  - Thời gian thực thi cần đủ nhanh
  - Phương pháp giải đã được nghiên cứu
  - Các bài toán trong các cuộc thi (ACM ICPC & IOI) có đặc trưng riêng
- 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)

1. Đọc và phân tích đề bài
    1. Đọc hiểu phần mô tả
    2. Xác nhận lại qua dữ liệu mẫu
2. Chọn giải thuật → CTDL
    1. Đoán nhận giải thuật qua ví dụ, thử nghiệm với dữ liệu nhỏ
    2. 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.
3. 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)

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

Ngôn ngữ lập trình

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

Bài tập về sử dụng Ngôn ngữ lập trình

Hãy làm mỗi bài tập trong không quá 10 dòng codes

1. Cho số nguyên *n* (*n* ≤ 15), hãy in *π* với *n* chữ số ở phần thập phân (sau khi đã làm tròn). (Ví dụ *n* = 2 → 3.14; *n* = 4 → 3.1416; *n* = 5 → 3.14159.)
2. Cho ngày tháng năm (dd/mm/yyyy), chỉ ra thứ trong tuần của ngày này (Thu Hai, Thu Ba,. . . , Chu Nhat). (Ví dụ ngày 09/08/2010 là Thu Hai.)
3. Cho *n* số nguyên (phát sinh ngẫu nhiên), hãy in mỗi giá trị một lần theo trình tự tăng dần.
4. Cho dãy gồm n bộ ba (DD, MM, YYYY) không trùng nhau là ngày sinh của n người, hãy sắp xếp dãy tăng dần theo tháng (MM), sau đó theo ngày (DD), và cuối cùng theo tuổi.

Bài tập về sử dụng Ngôn ngữ lập trình

5. Cho dãy *L* gồm các số nguyên đã được sắp tăng dần, số phần tử của *L* có thể lên đến 10^6. Hãy xác định giá trị v có trong *L* hay không khi chỉ dùng không quá 20 lượt so sánh.
6. Phát sinh tất cả các hoán vị của dãy gồm 10 chữ cái đầu tiên bảng chữ cái {'A', 'B', 'C', . . . , 'J'}.
7. Phát sinh tất cả các tập con của tập {0, 1, 2, . . . , *N*-1}, với *N* = 20.
8. Cho số nguyên trong cơ số X. Hãy cho giá trị của số trong có số Y ($2 \le X, Y \le 36$). Các số đều được biểu diễn ở dạng xâu ký tự. Ví dụ: "FF" trong cơ số 16 (hexadecimal) sẽ là "255" trong cơ số 10 (decimal) và là "11111111" trong cơ số 2 (binary).

Bài tập về sử dụng Ngôn ngữ lập trình

9. Cho chuỗi ký tự S. Từ đặc biệt là một từ gồm 2 ký số nằm sau 1 chữ cái in thường. Hãy thay thế tất cả những từ đặc biệt của S bởi 3 dấu sao "***". Ví dụ, với S = "Dong: a70 va z72 se duoc thay the, aa24 va a872 thi khong" sẽ được biến đổi thành: S = "Dong: *** va *** se duoc thay the, aa24 va a872 thi khong".
10. Cho 1 biểu thức toán học *E* với các phép toán cơ bản +, -, *, /. Hãy tính giá trị biểu thức. Ví dụ 3 + (8 - 7.5) * 10 / 5 - (2 + 5 * 7) sẽ cho giá trị -33.0.

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 trong các kỳ thi 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

Bài tập nhận dạng bài toán

1. Có n webpages ($1 \le n \le 10M$), webpage i có thứ hạng r_i. Cần lấy 10 trang có thứ hạng cao nhất. Hãy đề xuất thuật toán.
2. Cho danh sách L khoảng $10K$ số nguyên, truy vấn thường được hỏi: sum(i, j) = L[i] + L[i+1] + ...+ L[j]. Cấu trúc dữ liệu nào sẽ được dùng?
3. Cần tính độ dài đường đi ngắn nhất giữa 2 đỉnh trên đồ thị có hướng không có chu trình (DAG) với $|V|, |E| \le 100K$. Thuật toán nào sẽ được sử dụng với thời gian giới hạn thực thi là 3 giây?
4. Thuật toán nào để tìm được 10K số nguyên tố đầu tiên có độ phức tạp tính toán thấp nhất?

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

Worst AC Algorithm

N	Worst AC Algorithm	Ghi chú
≤ [10..11]	$O(n!)$, $O(n^6)$	Tổ hợp, hoán vị
≤ [15..18]	$O(2^n \times n^2)$	DP TSP
≤ [18..22]	$O(2^n \times n)$	DP with bitmask technique
≤ 100	$O(n^4)$	DP with 3 dimensions + $O(n)$
≤ 400	$O(n^3)$	Floyd Warshall's
≤ 2*K*	$O(n^2 \log_2 n)$	2-nested loops + a tree-related DS
≤ 10*K*	$O(n^2)$	Bubble/Selection/Insertion Sort
≤ 1*M*	$O(n \log_2 n)$	Merge Sort, building Segment Tree
≤ 100*M*	$O(n)$, $O(\log 2\ n)$, $O(1)$	Hầu hết các bài đều có $n \le 1M$ (I/O bottleneck)

Chạy thử & Sửa lỗi

Chương trình cần vượt qua bộ test của kỳ thi (được giữ bí mật)

- Dựng bộ test kiểm thử → xác định lỗi → sửa lỗi → kiểm thử
- Submit:
  - AC
  - Return code: (PE, WA, MTE, RTE, TLE) → đoán lỗi → dựng bộ test để xác nhận → sửa lỗi → submit

Lưu ý:

- Multies test case
- Các trường hợp ở biên dữ liệu (nhỏ/lớn nhất)
- Định dạng dữ liệu nhập/xuất

Một số trang online judge

1. http://uva.onlinejudge.org
2. http://codeforces.com
3. http://www.spoj.com
4. https://leetcode.com
5. https://icpcarchive.ecs.baylor.edu
6. http://www.topcoder.com
7. http://train.usaco.org

Tài liệu tham khảo

1. Robert Sedgewick, ***Algorithms.*** Addison Wesley, 1984 (1946)
2. Steven & Felix Halim, ***Competitive programming.*** Lulu Independent Publish, 2013
3. Nguyễn Thanh Tùng, ***Kỹ thuật Lập trình.*** 2017