Background

Tips
- Reading
- Planning
- Coding
- Timing
- Testing
- Debugging

Sample strategy

Summary

Contest Strategy

Robin Visser

IOI Training Camp
University of Cape Town

6 February 2016

Overview

1 Background

2 Tips
- Reading
- Planning
- Coding
- Timing
- Testing
- Debugging

3 Sample strategy

4 Summary

Background

- All good sportsmen spend considerable time planning an effective strategy. Programming contests are no exception.
- Having a strategy is an essential component to doing well in any olympiad contest.
- One's score is a combination of skill and adequate planning.
- One might have a different strategic approach for different contests.

Background

- All good sportsmen spend considerable time planning an effective strategy. Programming contests are no exception.
- Having a strategy is an essential component to doing well in any olympiad contest.
- One's score is a combination of skill and adequate planning.
- One might have a different strategic approach for different contests.

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

Background

- All good sportsmen spend considerable time planning an effective strategy. Programming contests are no exception.
- Having a strategy is an essential component to doing well in any olympiad contest.
- One's score is a combination of skill and adequate planning.
- One might have a different strategic approach for different contests.

Background

Tips
- Reading
- Planning
- Coding
- Timing
- Testing
- Debugging

Sample strategy

Summary

Background

- All good sportsmen spend considerable time planning an effective strategy. Programming contests are no exception.
- Having a strategy is an essential component to doing well in any olympiad contest.
- One's score is a combination of skill and adequate planning.
- One might have a different strategic approach for different contests.

Reading the questions

Background

Tips

Reading

Planning

Coding

Timing

Testing

Debugging

Sample strategy

Summary

- Always read all the questions before doing any coding. Many contests don't have the questions in order of difficulty.
- Read through each question thoroughly. Take note of any edge cases that may be easy to miss.
- Take note of the constraints, including subtasks.

Reading the questions

- Always read all the questions before doing any coding. Many contests don't have the questions in order of difficulty.
- Read through each question thoroughly. Take note of any edge cases that may be easy to miss.
- Take note of the constraints, including subtasks.

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

Reading the questions

- Always read all the questions before doing any coding. Many contests don't have the questions in order of difficulty.
- Read through each question thoroughly. Take note of any edge cases that may be easy to miss.
- Take note of the constraints, including subtasks.

Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

Planning

- Don't implement immediately after getting a possible solution.
- Consider the time complexity and memory of your solution. Think of optimisations to make.
- Implement the simplest possible solution. Don't over-complicate things.
- For a more algorithmically complex solution, try to judge if you can efficiently implement said solution. If not, perhaps go for a slower approach, but easier to implement.
- Try to judge how long a solution takes to implement compared to the pay-off in potential marks gained.

Planning

Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Don't implement immediately after getting a possible solution.
- Consider the time complexity and memory of your solution. Think of optimisations to make.
- Implement the simplest possible solution. Don't over-complicate things.
- For a more algorithmically complex solution, try to judge if you can efficiently implement said solution. If not, perhaps go for a slower approach, but easier to implement.
- Try to judge how long a solution takes to implement compared to the pay-off in potential marks gained.

Planning

Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Don't implement immediately after getting a possible solution.
- Consider the time complexity and memory of your solution. Think of optimisations to make.
- Implement the simplest possible solution. Don't over-complicate things.
- For a more algorithmically complex solution, try to judge if you can efficiently implement said solution. If not, perhaps go for a slower approach, but easier to implement.
- Try to judge how long a solution takes to implement compared to the pay-off in potential marks gained.

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

Planning

- Don't implement immediately after getting a possible solution.
- Consider the time complexity and memory of your solution. Think of optimisations to make.
- Implement the simplest possible solution. Don't over-complicate things.
- For a more algorithmically complex solution, try to judge if you can efficiently implement said solution. If not, perhaps go for a slower approach, but easier to implement.
- Try to judge how long a solution takes to implement compared to the pay-off in potential marks gained.

Planning

Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Don't implement immediately after getting a possible solution.
- Consider the time complexity and memory of your solution. Think of optimisations to make.
- Implement the simplest possible solution. Don't over-complicate things.
- For a more algorithmically complex solution, try to judge if you can efficiently implement said solution. If not, perhaps go for a slower approach, but easier to implement.
- Try to judge how long a solution takes to implement compared to the pay-off in potential marks gained.

Background

Tips
- Reading
- Planning
- Coding
- Timing
- Testing
- Debugging

Sample strategy

Summary

Coding

- Keep the code as simple as possible.
- Consider memorising a template to use shortcuts.
- Do not try to be too clever at the expense of wasting time or introducing bugs.
- Consider coding up brute force solutions.
- Don't be afraid to use white space, comments, meaningful variable names. Will make debugging much easier.
- Partial marks are your friend. Don't just try to code up one problem 100% at the expense of not getting any partials for other problems.

Coding

Contest Strategy
Robin Visser
Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Keep the code as simple as possible.
- Consider memorising a template to use shortcuts.
- Do not try to be too clever at the expense of wasting time or introducing bugs.
- Consider coding up brute force solutions.
- Don't be afraid to use white space, comments, meaningful variable names. Will make debugging much easier.
- Partial marks are your friend. Don't just try to code up one problem 100% at the expense of not getting any partials for other problems.

Coding

- Keep the code as simple as possible.
- Consider memorising a template to use shortcuts.
- Do not try to be too clever at the expense of wasting time or introducing bugs.
- Consider coding up brute force solutions.
- Don't be afraid to use white space, comments, meaningful variable names. Will make debugging much easier.
- Partial marks are your friend. Don't just try to code up one problem 100% at the expense of not getting any partials for other problems.

Coding

Contest Strategy
Robin Visser
Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Keep the code as simple as possible.
- Consider memorising a template to use shortcuts.
- Do not try to be too clever at the expense of wasting time or introducing bugs.
- Consider coding up brute force solutions.
- Don't be afraid to use white space, comments, meaningful variable names. Will make debugging much easier.
- Partial marks are your friend. Don't just try to code up one problem 100% at the expense of not getting any partials for other problems.

Coding

- Keep the code as simple as possible.
- Consider memorising a template to use shortcuts.
- Do not try to be too clever at the expense of wasting time or introducing bugs.
- Consider coding up brute force solutions.
- Don't be afraid to use white space, comments, meaningful variable names. Will make debugging much easier.
- Partial marks are your friend. Don't just try to code up one problem 100% at the expense of not getting any partials for other problems.

Coding

- Keep the code as simple as possible.
- Consider memorising a template to use shortcuts.
- Do not try to be too clever at the expense of wasting time or introducing bugs.
- Consider coding up brute force solutions.
- Don't be afraid to use white space, comments, meaningful variable names. Will make debugging much easier.
- Partial marks are your friend. Don't just try to code up one problem 100% at the expense of not getting any partials for other problems.

Timing

Contest Strategy
Robin Visser
Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Modern processors can handle roughly 10^8 to 10^9 operations per second.
- If $N \leq 10000$ then O(n^2), for $N \leq 500$ then O(n^3).
- If N very small ($N \leq 20$) then try brute force.
- Don't optimise more than what's required (e.g. going from a O$(n \log n)$ solution to a O(n) solution is probably not necessary)

Timing

Contest Strategy

Robin Visser

Background

Tips
- Reading
- Planning
- Coding
- Timing
- Testing
- Debugging

Sample strategy

Summary

- Modern processors can handle roughly 10^8 to 10^9 operations per second.
- If $N \leq 10000$ then O(n^2), for $N \leq 500$ then O(n^3).
- If N very small ($N \leq 20$) then try brute force.
- Don't optimise more than what's required (e.g. going from a O($n \log n$) solution to a O(n) solution is probably not necessary)

Timing

- Modern processors can handle roughly 10^8 to 10^9 operations per second.
- If $N \leq 10000$ then O(n^2), for $N \leq 500$ then O(n^3).
- If N very small ($N \leq 20$) then try brute force.
- Don't optimise more than what's required (e.g. going from a O($n \log n$) solution to a O(n) solution is probably not necessary)

Timing

- Modern processors can handle roughly 10^8 to 10^9 operations per second.
- If $N \leq 10000$ then O(n^2), for $N \leq 500$ then O(n^3).
- If N very small ($N \leq 20$) then try brute force.
- Don't optimise more than what's required (e.g. going from a O($n \log n$) solution to a O(n) solution is probably not necessary)

Testing

- Make up unit test cases to check your solution, other than the sample test cases given to you.
- Consider boundary cases and special cases (small/large values, off-by-one errors)
- Use brute force solutions to compare test data with your optimised solutions.
- Use assertions: `assert(condition);`
- Most people don't spend enough time testing, although time spent on testing will depend on whether detailed feedback is available.

Testing

- Make up unit test cases to check your solution, other than the sample test cases given to you.
- Consider boundary cases and special cases (small/large values, off-by-one errors)
- Use brute force solutions to compare test data with your optimised solutions.
- Use assertions: `assert(condition);`
- Most people don't spend enough time testing, although time spent on testing will depend on whether detailed feedback is available.

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

Testing

- Make up unit test cases to check your solution, other than the sample test cases given to you.
- Consider boundary cases and special cases (small/large values, off-by-one errors)
- Use brute force solutions to compare test data with your optimised solutions.
- Use assertions: `assert(condition);`
- Most people don't spend enough time testing, although time spent on testing will depend on whether detailed feedback is available.

Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

Testing

- Make up unit test cases to check your solution, other than the sample test cases given to you.
- Consider boundary cases and special cases (small/large values, off-by-one errors)
- Use brute force solutions to compare test data with your optimised solutions.
- Use assertions: `assert(condition);`
- Most people don't spend enough time testing, although time spent on testing will depend on whether detailed feedback is available.

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

Testing

- Make up unit test cases to check your solution, other than the sample test cases given to you.
- Consider boundary cases and special cases (small/large values, off-by-one errors)
- Use brute force solutions to compare test data with your optimised solutions.
- Use assertions: `assert(condition);`
- Most people don't spend enough time testing, although time spent on testing will depend on whether detailed feedback is available.

Debugging

Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Simplest way to debug is to print additional output (trace statements)
- Can often be the easiest way to quickly debug a small error in the code.
- For more advanced debugging, the GDB Debugger (gdb) is very useful

Debugging

Contest Strategy
Robin Visser
Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

- Simplest way to debug is to print additional output (trace statements)
- Can often be the easiest way to quickly debug a small error in the code.
- For more advanced debugging, the GDB Debugger (gdb) is very useful

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

Debugging

- Simplest way to debug is to print additional output (trace statements)
- Can often be the easiest way to quickly debug a small error in the code.
- For more advanced debugging, the GDB Debugger (gdb) is very useful

Strategy for COCI

Contest Strategy

Robin Visser

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

What would be the best strategy to approach a COCI contest?

- 3 hours with 6 problems
- Partial scoring.
- Detailed feedback?
- Only 30 minutes per question. Either try for partials for all questions, or solve a few perfectly?

Strategy for COCI

What would be the best strategy to approach a COCI contest?

- 3 hours with 6 problems
- Partial scoring.
- Detailed feedback?
- Only 30 minutes per question. Either try for partials for all questions, or solve a few perfectly?

Strategy for COCI

Contest Strategy
Robin Visser
Background
Tips
Reading
Planning
Coding
Timing
Testing
Debugging
Sample strategy
Summary

What would be the best strategy to approach a COCI contest?

- 3 hours with 6 problems
- Partial scoring.
- Detailed feedback?
- Only 30 minutes per question. Either try for partials for all questions, or solve a few perfectly?

Strategy for COCI

What would be the best strategy to approach a COCI contest?

- 3 hours with 6 problems
- Partial scoring.
- Detailed feedback?
- Only 30 minutes per question. Either try for partials for all questions, or solve a few perfectly?

Background

Tips
- Reading
- Planning
- Coding
- Timing
- Testing
- Debugging

Sample strategy

Summary

Strategy for COCI

What would be the best strategy to approach a COCI contest?

- 3 hours with 6 problems
- Partial scoring.
- Detailed feedback?
- Only 30 minutes per question. Either try for partials for all questions, or solve a few perfectly?

Background

Tips
- Reading
- Planning
- Coding
- Timing
- Testing
- Debugging

Sample strategy

Summary

Summary

- Every person has their own strategy that works for them.
- The only way to determine what works best for you is to practice contests regularly.
- Don't be afraid to try new approaches when practicing at home (you don't want to adopt a completely new strategy only at the IOI)

Background

Tips
Reading
Planning
Coding
Timing
Testing
Debugging

Sample strategy

Summary

Summary

- Every person has their own strategy that works for them.
- The only way to determine what works best for you is to practice contests regularly.
- Don't be afraid to try new approaches when practicing at home (you don't want to adopt a completely new strategy only at the IOI)

Summary

- Every person has their own strategy that works for them.
- The only way to determine what works best for you is to practice contests regularly.
- Don't be afraid to try new approaches when practicing at home (you don't want to adopt a completely new strategy only at the IOI)