

1. Introduction to the judge environment

(by means of a sample problem)

2. Prefix sum/precomputation technique

Sample Problem: Even Pairs

Input:

- ▶ first line: a positive integer n
- ▶ second line: a sequence $x_0, \dots, x_{n-1} \in \{0, 1\}$

Output: a single line containing the **number of pairs** $0 \leq i \leq j < n$ such that

$$x_i + \dots + x_j$$

is even.

Example

Input: $n = 4$

$A :$ 0 1 1 1

Output: # of 'even pairs': 0

Example

Input: $n = 4$

$A :$ 0 1 1 1

Interval (red) sum = 0

Output: # of 'even pairs': 1

Example

Input: $n = 4$

$A :$ 0 1 1 1

Interval (red) sum = 1

Output: # of 'even pairs': 1

Example

Input: $n = 4$

$A :$ 0 1 1 1

Interval (red) sum = 2

Output: # of 'even pairs': 2

Example

Input: $n = 4$

$A :$ 0 1 1 1

Interval (red) sum = 3

Output: # of 'even pairs': 2

Example

Input: $n = 4$

$A :$ 0 1 1 1

Interval (red) sum = 2

Output: # of 'even pairs': 3

Example

Input: $n = 4$

$A :$ 0 1 1 1

Interval (red) sum = 2

Output: # of 'even pairs': 4

First Approach

- (1) for all pairs $i \leq j$, compute the sum $x_i + \dots + x_j$
- (2) if it is even, increment a counting variable

A few points (**good!**) but... also a **TIMELIMIT** error on harder test sets!

- ▶ this means that our algorithm is too slow
- ▶ we have **three** nested loops: **two** for going over all pairs $i \leq j$, and **one** for summing up the sequence $x_i + \dots + x_j$
- ▶ this type of analysis is **very important** in this course

Running time: $O(n^3)$

How can we improve?

$$\underbrace{x_0, x_1, \dots, x_{i-1}, x_i, \dots, x_j}_{S_j}, x_{j+1}, \dots, x_{n-1}$$

$$\underbrace{x_0, x_1, \dots, x_{i-1}}_{S_{i-1}}, x_i, \dots, x_j, x_{j+1}, \dots, x_{n-1}$$

The sum $x_i + \dots + x_j$ can be represented as

$$S_j - S_{i-1}$$

Second Approach

Observation:

$$\begin{aligned}x_i + \cdots + x_j &= \sum_{a=0}^j x_a - \sum_{b=0}^{i-1} x_b \\ &= S_j - S_{i-1}\end{aligned}$$

(1) calculate **partial sums** $S_i = \sum_{a=0}^i x_a$ in one iteration

$$\begin{array}{rcccc}A : & 0 & 1 & 1 & 1 \\S : & 0 & 1 & 2 & 3\end{array}$$

(2) for every $i \leq j$ check the parity of $S_j - S_{i-1}$

Running time: $O(n^2)$

Third Approach

Observation:

$$\begin{aligned}x_i + \cdots + x_j &= \sum_{a=0}^j x_a - \sum_{b=0}^{i-1} x_b \\ &= S_j - S_{i-1}\end{aligned}$$

How can possibly $x_i + \cdots + x_j$ be even based **just on** S_j and S_{i-1} ?! Only if **both** S_j, S_{i-1} are **even** or **both** S_j, S_{i-1} are **odd**!

- ▶ for every $i \leq j$ check the parity of $S_j - S_{i-1}$ **WASTEFUL!**

Third Approach

Observation:

$$\begin{aligned}x_i + \cdots + x_j &= \sum_{a=0}^j x_a - \sum_{b=0}^{i-1} x_b \\&= S_j - S_{i-1}\end{aligned}$$

- (1) calculate **partial sums** $S_i = \sum_{a=0}^i x_a$ in one iteration
- (2) $E = \#$ of S_i that are even
- (3) $O = \#$ of S_i that are odd
- (4) the result is:

$$\begin{array}{ccccc}\binom{E}{2} & + & \binom{O}{2} & + & E \\ \text{pairs of } S_j, S_{i-1} \text{ even} & & \text{pairs of } S_j, S_{i-1} \text{ odd} & & \text{sums of the form } x_0 + \cdots + x_i\end{array}$$

Running time: $O(n)$

Technique: Partial Sums/Precomputing

- ▶ Precomputing partial sums allows computing the sum of the elements in an interval in **constant time**.
- ▶ More generally, precomputing certain values can speed up the running time of an algorithm.

Judge Feedback

Besides **CORRECT**, **TIMELIMIT**, and **WRONG-ANSWER**, the judge can give the following feedback:

ASSERTION-FAILURE SIGABRT: memory screw-up or assertion failure

SEGMENTATION-FAULT SIGSEGV: memory screw-up (e.g. out-of-bounds)

RUN-ERROR nonzero exit status

FORBIDDEN bad syscall or other safety

The forum is your main tool for discussing ideas and getting help. Use it.

Of course, you will only learn if you first try to solve the problems **on your own**.

Forum: How To Ask Questions

1. Apply spoiler warnings
2. Describe the problem, not your guesses or summaries
3. Code: describe what fails and what you expect instead
4. Code: post **minimal** examples
5. Do not rush to claim that you have found a bug

Example

SPOILER<<<

Set this text to have a **white foreground**. It will then be invisible unless marked. The <<< ... >>> exploit a bug in the email plugin to also remove the text in plain-text email.

>>>

Forum: How To Ask Questions

1. Apply spoiler warnings
2. Describe the problem, not your guesses or summaries
3. Code: describe what fails and what you expect instead
4. Code: post **minimal** examples
5. Do not rush to claim that you have found a bug

Example

Bad When I compile, it tells me it cannot find it.

Good When I run `g++ -o foo foo.cpp`, I get
`bash: $'g++\302\240-o': command not found`

Forum: How To Ask Questions

1. Apply spoiler warnings
2. Describe the problem, not your guesses or summaries
3. Code: describe what fails and what you expect instead
4. Code: post **minimal** examples
5. Do not rush to claim that you have found a bug

Example

Bad The code below doesn't work. Help?

Good I am trying to solve Problem 1. I tried strategy **something**. My code is below. For some reason, when running it on the provided test case it emits **no solution** instead of **1**. What am I doing wrong?

Forum: How To Ask Questions

1. Apply spoiler warnings
2. Describe the problem, not your guesses or summaries
3. Code: describe what fails and what you expect instead
4. Code: post **minimal** examples
5. Do not rush to claim that you have found a bug

Example

Bad When I call `.foo()` on a **vector**, it segfaults. Bug!

Good I am trying to **something**. The code is below. I get a segfault in the line that calls `.foo()`, but if I remove that line the program continues. What am I doing wrong?

Learning C++ is beyond the scope of this course.

- ▶ True beginners should probably read a book
- ▶ People familiar with the syntactic family (e.g. C, Java, C#) may get away with a tutorial
- ▶ Useful in any case: [C++ FAQ Lite](#)
- ▶ **NEW!** Updated version of the '[A Short Introduction to C++ for the Algorithms Lab](#)'

NEW! Updated version of the [‘A Short Introduction to C++ for the Algorithms Lab’](#)

Contains:

- ▶ Easy exercises to get you started with C++ including Judge feedback
- ▶ Handling I/O
- ▶ Fundamental data types and basic concepts of the C++ standard library
- ▶ How to compile, run, and debug

Does not contain:

- ▶ Introduction to C++ programming
- ▶ Control structures
- ▶ Memory management
- ▶ Objected Oriented Programming in C++

It is important that you know certain parts of the C++ standard library, such as:

- ▶ how to do I/O using `<iostream>`,
- ▶ how and when to use which container: `vector`, `set`, `map`, `stack`, `queue`, and `priority_queue`,
- ▶ how to use the `sort` function from the `<algorithm>` header file, and
- ▶ how to use iterators.

IMPORTANT! Read the '[A Short Introduction to C++ for the Algorithms Lab](#)' document and familiarise yourselves with the concepts within.

That's it!

The most important things to remember:

- ▶ You can find all that you need at our website:

<https://www.cadmo.ethz.ch/education/lectures/HS20/algolab/index.html>

- ▶ For questions, you should use the forum:

<https://moodle-app2.let.ethz.ch/course/view.php?id=13169>

- ▶ Today we will publish the first week's exercises on Moodle & [CodeExpert](#)

- ▶ The public input/output files are available on [CodeExpert](#) ('Download Project' to have them locally)

- ▶ Today we will also publish easy exercises to help you get started with C++.

- ▶ Next Monday at 14:00, we have the first **Problem of The Week!**