Competitive Programming and Mathematics Society

Programming Workshop #0

Introduction to Programming Problems

Isaiah Iliffe

Welcome

CPMSOC

- There will be a programming workshop every two weeks.
- Each workshop will run for up to two hours.
- Each workshop (including this one) will have an accompanying problem set, where you can practise what has been taught.
- Please feel free to ask questions at any time.

Today's Workshop

CPMSOC

1. Welcome
2. Today's Workshop
3. What is a programming problem?
4. Why solve problems?
5. Competitions and training
6. Your First Submission
7. Prefix Sums
8. Time Complexity
9. Group Activity
10. Wrap up

What is a programming problem?

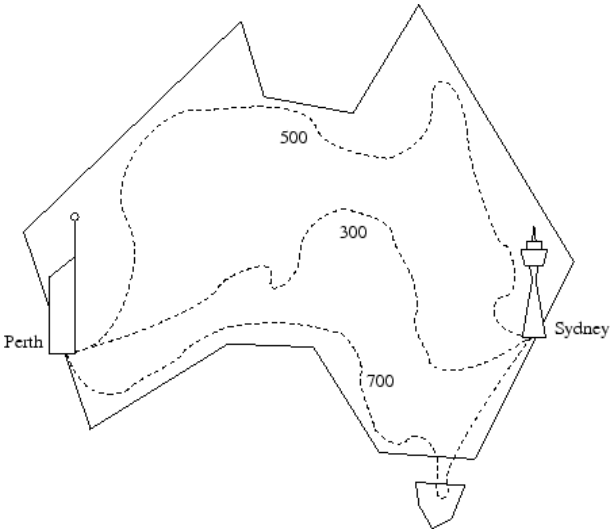
CPMSOC

In this scenario, there are three different routes between Perth and Sydney.

Each route has a different number of customers who will buy your encyclopedias.

You are planning a round trip from Perth to Sydney to Perth. What is the greatest number of different customers you can reach along the way?

Source: AIO 2009



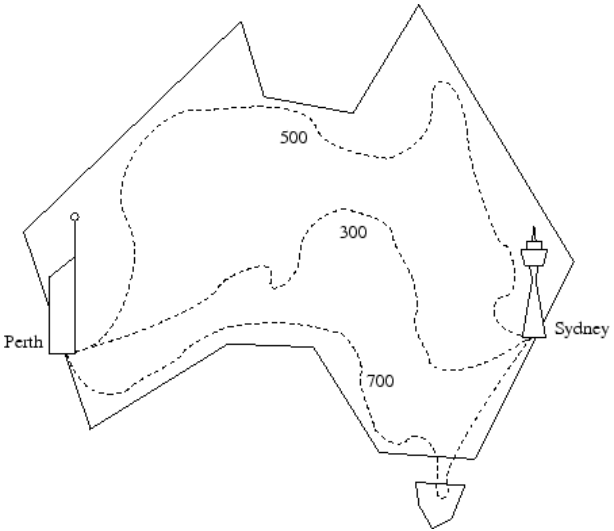
What is a programming problem?

CPMSOC

That was a long way of saying: given 3 numbers, find the sum of the 2 largest ones.

Our task is to write a program that does this for any 3 numbers given to it.

In a competition, we submit this program and a judge checks whether for a bunch of hidden inputs, your code outputs the right answer. From this, the judge will have a pretty good idea about whether your program works.



Why solve problems?

CPMSOC

- To learn, practise and enjoy problem solving, algorithms and data structures
- To practise your implementation and general programming skills
- To prepare for programming competitions and interviews

Competitions and training

CPMSOC

- International Collegiate Programming Contest
  - Divisionals/Regionals/World Finals
  - ANZAC League
- Big Companies
  - Google Code Jam/Kickstart
  - Facebook Hacker Cup
  - Amazalgo
- Society Competitions
  - CSESoc + Industry sponsors
  - CPMSoc (hopefully)

- Online platforms (for training and/or regular contests)
  - AtCoder
  - Codeforces
  - Topcoder
  - CodeChef
  - ORAC
  - USACO
  - Project Euler
  - HackerRank
  - LeetCode
  - Probably more...

Your First Submission

CPMSOC

- **Statement** Dallas and Jeffrey want to buy pizza for a CPMSoc event, but they have no money. Tom lends A dollars to Dallas and B dollars to Jeffrey from the fat stacks of cash he makes as a front end developer. How much did Tom lend in total?
- **Input** Two integers, A and B ($0 \leq A, B \leq 10$).
- **Output** A single integer, the total amount Tom lent to Dallas and Jeffrey in dollars.

Your First Submission

CPMSOC

- **Statement** Dallas and Jeffrey want to buy pizza for a CPMSoc event, but they have no money. Tom lends A dollars to Dallas and B dollars to Jeffrey from the fat stacks of cash he makes as a front end developer. How much did Tom lend in total?
- **Input** Two integers, A and B ($0 \leq A, B \leq 10$).
- **Output** A single integer, the total amount Tom lent to Dallas and Jeffrey in dollars.

- **What is the question asking?** Print $A + B$, given A and B.

Your First Submission

CPMSOC

Implementation (C or C++)

```c
#include <stdio.h>
int main() {
    // Read input
    int a, b;
    scanf("%d %d", &a, &b);

    // Compute answer
    int ans = a + b;

    // Print output
    printf("%d\n", ans);
}
```

Running locally

- Compile: g++ source.cpp
- Run: ./a.out
- Redirect input file to stdin: ./a.out < input.txt

Online shell

- Go to the website cpp.sh

Submitting

- Submit file or copy/paste code to the judging site.

Prefix Sums

Motivation

CPMSOC

Bob has an array of n numbers.

Alice comes up to Bob and asks him lots of questions of the form: What is the sum of the values from index l to index r?

Help Bob **quickly** deal with each of Alice's questions!

Theory

CPMSOC

Theorem (Fundamental Theorem of Prefix Sums)

For any sequence $a_1, a_2, ..., a_n$ and $1 \leq l \leq r \leq n$ it holds that

$$\sum_{i=l}^{r} a_i = \sum_{i=1}^{r} a_i - \sum_{i=1}^{l-1} a_i.$$

Theory

CPMSOC

Theorem (Fundamental Theorem of Prefix Sums)

For any sequence $a_1, a_2, ..., a_n$ and $1 \leq l \leq r \leq n$ it holds that

$$\sum_{i=l}^{r} a_i = \sum_{i=1}^{r} a_i - \sum_{i=1}^{l-1} a_i.$$

Proof.

By the associativity of addition,

$$\sum_{i=1}^{l-1} a_i + \sum_{i=l}^{r} a_i = \sum_{i=1}^{r} a_i.$$

The result follows upon rearrangement. ■

Code

CPMSOC

```
#include <stdio.h>
int N, A[100005], pre[100005], Q, l, r;
int main() {
    // Read input
    scanf("%d", &N);
    for (int i = 1; i <= N; i++) scanf("%d", &A[i]);
    // Compute prefix sums
    for (int i = 1; i <= N; i++) pre[i] = pre[i-1] + A[i];
    // Answer queries
    scanf("%d", &Q);
    for (int i = 1; i <= Q; i++) {
        scanf("%d %d", &l, &r);
        printf("%d\n", pre[r] - pre[l-1]);
    }
}
```

Time Complexity

Time Complexity

CPMSOC

Definition

- We say two algorithms have the same time complexity if, as the input parameters get large, the number of operations in one approaches a constant factor of the number of operations in the other.

Time Complexity

CPMSOC

Definition

- We say two algorithms have the same time complexity if, as the input parameters get large, the number of operations in one approaches a constant factor of the number of operations in the other.
- This allows us to compare algorithms meaningfully, while preserving the useful ambiguity of what constitutes an "operation".

Time Complexity

CPMSOC

Definition

- We say two algorithms have the same time complexity if, as the input parameters get large, the number of operations in one approaches a constant factor of the number of operations in the other.
- This allows us to compare algorithms meaningfully, while preserving the useful ambiguity of what constitutes an "operation".
- For example if we are printing all n elements in an array, we would call that an $O(n)$ algorithm as there are n operations. If we were printing them out 2, 3, 100, or 1000 times we would still consider that $O(n)$ as they are constant factors.

Time Complexity

CPMSOC

Definition

- We say two algorithms have the same time complexity if, as the input parameters get large, the number of operations in one approaches a constant factor of the number of operations in the other.
- This allows us to compare algorithms meaningfully, while preserving the useful ambiguity of what constitutes an "operation".
- For example if we are printing all n elements in an array, we would call that an $O(n)$ algorithm as there are n operations. If we were printing them out 2, 3, 100, or 1000 times we would still consider that $O(n)$ as they are constant factors.
- However, if we printed the whole array once for every element in the array, then we would be printing n elements n times each. We would call that $O(n^2)$.

Time Complexity

CPMSOC

Example

- Finding the maximum value in an array of n numbers can be performed in $O(n)$.
  Exercise: Design an algorithm to find the sum of the two largest values in an array (generalised version of today's first problem) in $O(n)$.

Time Complexity

CPMSOC

Example

- Finding the maximum value in an array of n numbers can be performed in $O(n)$.
  Exercise: Design an algorithm to find the sum of the two largest values in an array (generalised version of today's first problem) in $O(n)$.
- Comparison based sorting can be performed in $O(n \log n)$.

Time Complexity

CPMSOC

Example

- Finding the maximum value in an array of n numbers can be performed in $O(n)$.
  Exercise: Design an algorithm to find the sum of the two largest values in an array (generalised version of today's first problem) in $O(n)$.
- Comparison based sorting can be performed in $O(n \log n)$.
- 3SUM can be solved in $O(n^2)$.
  Exercise: Is there an algorithm to solve the 3SUM problem in time $O(n^{2-\epsilon})$, for some $\epsilon > 0$?.

Magic Numbers

CPMSOC

Definition

As a very rough guide, a reasonable judging machine can perform 50 to 200 million medium sized operations in a second.

Magic Numbers

CPMSOC

Definition

As a very rough guide, a reasonable judging machine can perform 50 to 200 million medium sized operations in a second.

Maximum n	Appropriate time complexity
10	$O(n!)$
20	$O(n \cdot 2^n)$
100	$O(n^3)$
10^3	$O(n^2)$
10^5	$O(n^{3/2})$ or $O(n \log^2 n)$
10^6	$O(n \log n)$
10^7	$O(n)$
10^9+	$O(\log n)$ or $O(1)$

Problem – Bad Subarrays

CPMSOC

You are given an array of N integers. Alice thinks the number 13 is bad. Determine how many distinct subarrays (a list of values $a_l, a_{l+1}, a_{l+2}, ..., a_r$) have a sum of 13.

First, solve this in $O(N^3)$ or $O(N^2)$.

Problem – Bad Subarrays

CPMSOC

You are given an array of N integers. Alice thinks the number 13 is bad. Determine how many distinct subarrays (a list of values $a_l, a_{l+1}, a_{l+2}, ..., a_r$) have a sum of 13.

First, solve this in $O(N^3)$ or $O(N^2)$.

Now, solve this in $O(N)$.

Group Activity

Group Activity Problem – Walkscotch

CPMSOC

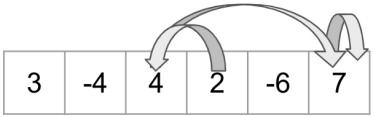
Bob is playing Walkscotch on a row of N squares, with the ith square having an integer point value of A_i.

He starts on square M with score 0. In one move, he walks to another square, or stays still. For each square between the start and finish square inclusive, the point value of that square is added to Bob's score.

Bob will perform K moves. Determine the maximum score he can achieve.

Constraints: $1 \leq M \leq N \leq 10^5, K \leq 10^9, |A_i| \leq 10^4$

Sample Input/Output:



N=6, M=4, K=3
Answer = (2+4)+(4+2-6+7)+(7)=20

Subtasks (easier versions of the problem if you get stuck):
(1) $N = 1$ (2) $M = 1$ and $K = 1$ (3) $K = 1$ (4) All $A_i \geq 0$ (5) All $A_i \leq 0$ (6) $K = 2$ (7) $M = 1$

Wrap up

CPMSOC

- See the problem set at vjudge.net/group/cpmsoc, so you can practise what has been taught.
  - Addition
  - Prefix sum queries
  - Good subarrays (similar to bad subarrays from today)
  - IOI 2005 Garden (Extension)
- Feel free to ask for help or to discuss problems on the Discord.
- Maths workshop 6:30PM-8:30PM next week.