Introduction
Complete Search

Definition?

Problem solving is a set of steps and processes to be done to reach to output

Problem solving Steps:

- Problem Definition : identify on output and input and arithmetic and logic operation to be done
- Algorithm preparation : is one of method used to solve problem
- Program design : Translate the flowchart to programming languages to solve it in computer
- Program testing : discover program errors and correct them

What is the competitive programming?

Timing

Typing

Teamwork

Why should we learn competitive programming?

facebook HACKERCUP

code jam
print "hello, world!"

icpc.foundation
icpc

IEEEXTREME PROGRAMMING COMPETITION

EOI
الأولمبياد المصري للمعلوماتية
EGYPTIAN OLYMPIAD IN INFORMATICS

How to be ready?

AtCoder

hackerearth

HackerRank

CODEFORCES

Time Complixety

1

The efficiency of algorithms is important in competitive programming

Time Complexity of an algorithm estimates how much time the algorithm will use for some input.

The time complexity of an algorithm is denoted $O(\ldots)$ where the three dots represent some function. Usually, the variable n denotes the input size. For example, if the input is an array of numbers, n will be the size of the array, and if the input is a string, n will be the length of the string.

Calculation Rules

Loops: A common reason why an algorithm is slow is that it contains many loops that go through the input

the time complexity of the following code is $O(n)$:

```
for (int i = 1; i <= n; i++)
{
        // code
}
```

And the time complexity of the following code is $O(n^2)$:

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++)
{
        // code
    }
}
```

Calculation Rules

Order of Magnitude : A time complexity does not tell us the exact number of times the code inside a loop is executed, but it only shows the order of magnitude. In the following examples, the code inside the loop is executed $3n$, $n + 5$, and $n/2$ times, but the time complexity of each code is $O(n)$.

```
for (int i = 1; i <= n+5; i++) {
        // code
}
```

```
for (int i = 1; i <= 3*n; i++) {
        // code
}
```

Calculation Rules

Phases: If the algorithm consists of consecutive phases, the total time complexity is the largest time complexity of a single phase.

For example, the following code consists of three phases with time complexities $O(n), O(n^2)$ and $O(n)$. Thus, the total time complexity is $O(n + n^2 + n) \approx O(n^2)$.

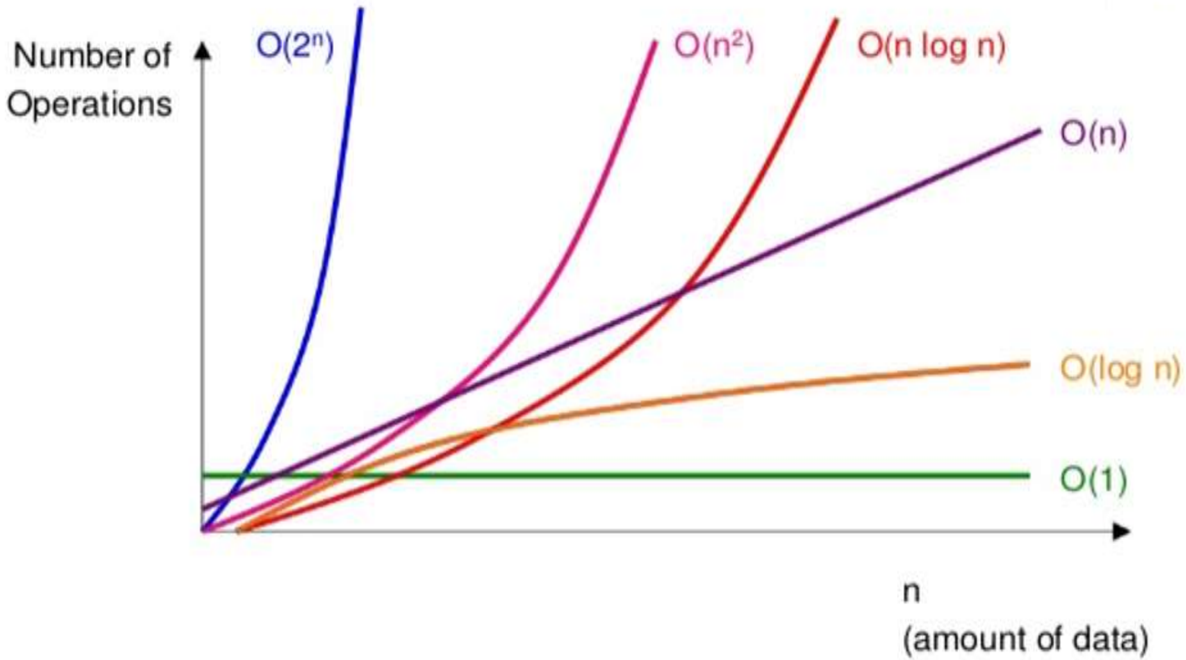
```
for (int i = 1; i <= n; i++) { // code }

for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) { //
code}
 }

for (int i = 1; i <= n; i++) { // code}
```

Calculation Rules

Complexity classes



Calculation Rules

input size	required time complexity
$n \leq 10$	$O(n!)$
$n \leq 20$	$O(2^n)$
$n \leq 500$	$O(n^3)$
$n \leq 10^4$	$O(n^2)$
$n \leq 10^5$	$O(N)$ or $O(N \log N)$
$n \leq 10^6$	$O(N)$

Activity!

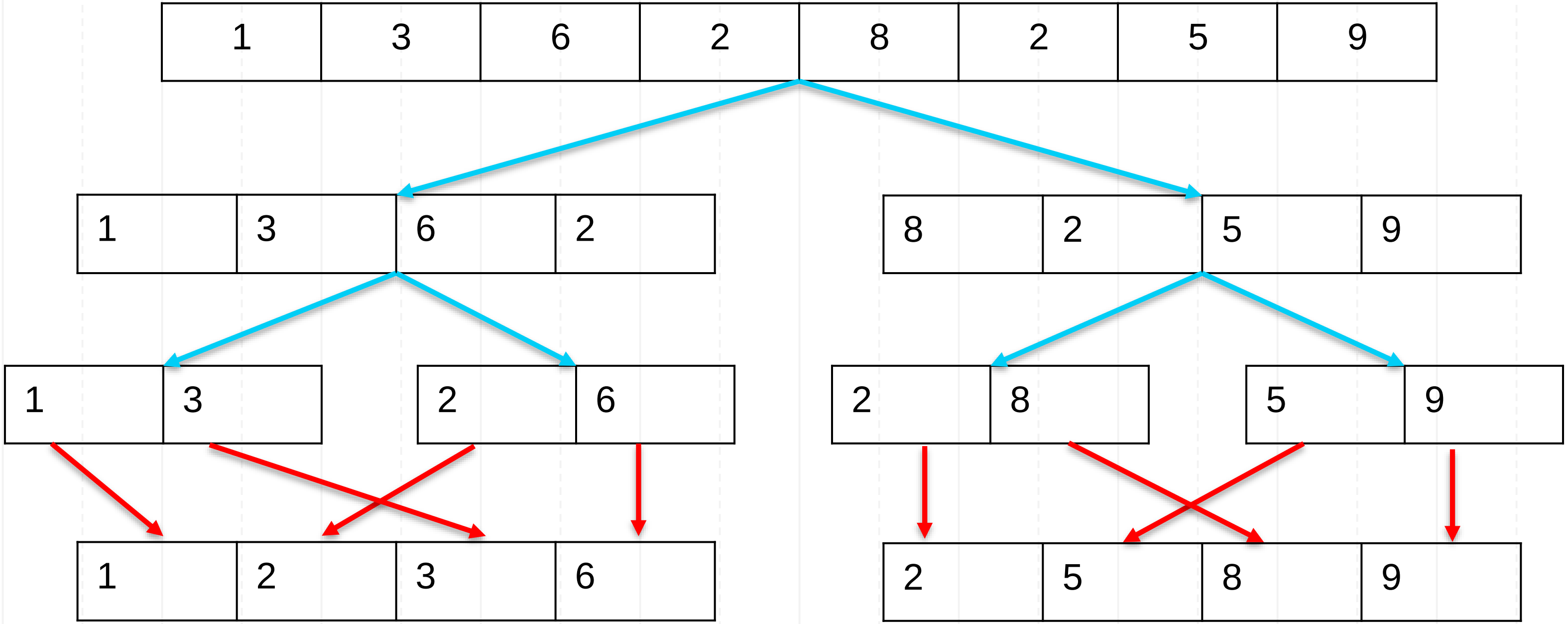
2 Sorting

Many efficient algorithms use sorting as a subroutine

Sort Algorithms

Algorithm technique	time complexity
bubble sort	$O(n^2)$
insertion sort	$O(n^2)$
selection sort	$O(n^2)$
quick sort	$O(N \log N)$ / $O(n^2)$
Merge sort	$O(N \log N)$

Merge sort



Built-in Functions

0ffx55 – 0ffx50 = 5

Memory	0ffx50	0ffx51	0ffx52	0ffx53	0ffx54	0ffx55
Index	0	1	2	3	4	5
Value	1	3	12	8	3	2

Function uses memory pointer, so I have to pass first pointer and last pointer

```
sort( arr , arr + n )
```

Practice Time

You have set of N element and print them sorted and unique (no element iterates more than once) *Use Sorting*

Input	Output
8 3 5 3 3 1 2 1 2	1 2 3 5

Built-in Functions

min ⇩ (0ffx50) max ⇩ (0ffx52)

Memory	0ffx50	0ffx51	0ffx52	0ffx53	0ffx54	0ffx55
Value	1	3	12	8	3	2

Function uses memory pointer, so function will return pointer to memory cell

```
*max_element( arr , arr + n )
*min_element( arr , arr + n )
reverse( arr , arr + n )
```

$O(N)$

3 Mathematics

it is not possible to become a successful CP without having good mathematical skills.

Sum Formula

1	2	3	4	5	6	7	8	9

$$1 + 9 = 10$$

1	2	3	4	5	6	7	8	9

$$2 + 8 = 10$$

Sum Formula

1	2	3	4	5	6	7	8	9

$$3 + 7 = 10$$

1	2	3	4	5	6	7	8	9

$$4 + 6 = 10$$

Sum Formula

We noticed that each time sum between last and first is $N + 1$

So, we have $\frac{N}{2}$ pair that sum is $(N + 1)$ which mean:

$$\sum_{x=1}^{x=n} x = 1 + 2 + 3 + 4 \ldots\ldots = \frac{N * (N + 1)}{2}$$

Sum of odd Formula

We noticed that the sum between odd values equal number of power 2

values	sum
1 + 3	$4 = 2^2$
1+ 3 + 5	$9 = 3^2$
1 + 3 + 5 + 7	$16 = 4^2$
1 + 3 + 5 + 7+…	n^2

The sum formula of odd values $[\ 1\ , N\ ]\ is\ N^2$

Sum of even Formula

We noticed that the sum between odd values equal number of power 2

values	sum
2 + 4	6 = 2 $*$ (2 + 1)
2 + 4 + 6	12 = 3 $*$ (3 + 1)
2 + 4 + 6 + 8	20 = 4 $*$ (4 + 1)
2 + 4 + 6 + 8+…	$N * (N + 1)$

The sum formula of even values $[1, N]$ is $N * (N + 1)$

Greedy Algorithms

4

Greedy algorithms is strategy that making best choice at this moment

Greedy Strategy

A **greedy algorithm** constructs a solution to the problem by always making a choice that looks the best at the moment. A greedy algorithm never takes back its choices, but directly constructs the final solution. For this reason, greedy algorithms are usually very efficient.

The difficulty in designing greedy algorithms is to find a greedy strategy that always produces an optimal solution to the problem.

Problems

https://codeforces.com/problemset/problem/888/B
https://codeforces.com/problemset/problem/785/B
https://codeforces.com/problemset/problem/545/D
https://codeforces.com/contest/545/problem/B
https://codeforces.com/contest/270/problem/B

THANKS!

Any questions?