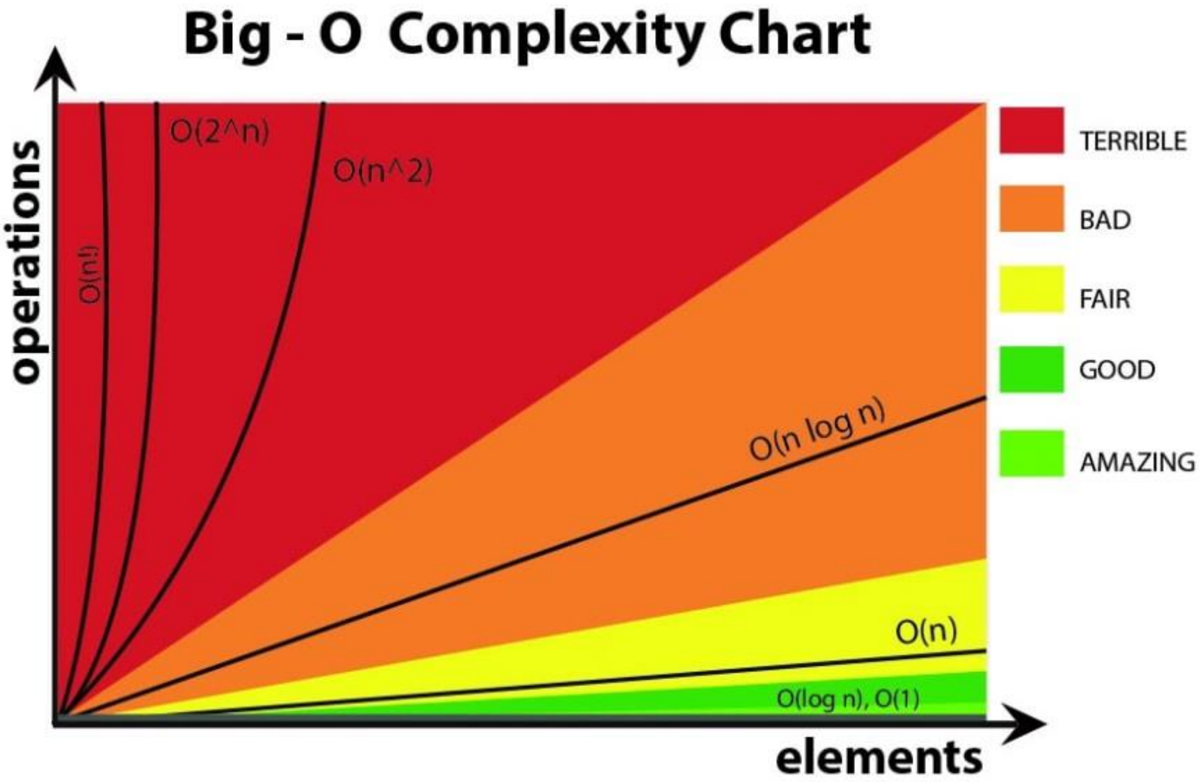
কমপ্লেক্সিটি – ও বিগ o নোটেশন উদাহরণ সহ এর ব্যাখ্যা



কমপ্লেক্সিটি :

আমরা যখন একটি সমস্যাকে সমাধানের জন্য কোন অ্যালগরিদম সব থেকে বেশী কার্যকারী এবং কেন কার্যকারী তা খুঁজে পেতে হলে আমাকে অবশ্যই আমাকে কমপ্লেক্সিটি সম্পর্কে সম্পূর্ণ ধারণা থাকা লাগবে। নিচে টাইম কমপ্লেক্সিটি ও স্পেস কমপ্লেক্সিটি নিয়ে বিষদ ভাবে আলোচনা করা হলো ।

টাইম কমপ্লেক্সিটি :

মনে করুন আপনি একটি প্রোগ্রাম লিখলেন। এখন আপনার প্রোগ্রামটি রান হতে যে সময় নিবে সেটি গণিতের মাধ্যমে বের করাই হলো টাইম কমপ্লেক্সিটি । আমারা কোন কাজ , বিশেষ করে যদি বলি গননার কাজ কেন নিজে না করে কম্পিউটারকে দিয়ে করাই ! ভেবে দেখেছি কখনো ? মনে করেন, ১ কোটি সংখ্যা যদি আপনাকে যোগ করতে দেওয়া হয় কত দিন সময় লাগবে এক বার একটু চিন্তা করে দেখেন তো ? আমার তো মনে হয় কয়েক দিন লেগে যাবে আপনিও মনে হয় সেটাই বলবেন । আর যদি আমরা এই কাজ টা কম্পিউটারকে দিয়ে করাই কত সহজে হয় কত অল্প সময়ে কাজ টা করা যাবে হয়তো ১ সেকেন্ডের ও অনেক কম সময় লাগবে । তাই কাজটি করতে কত সময় লেগেছে এটাই কিন্তু মুখ্য এখানে , যদি কম্পিউটারেরও অনেক সময় লাগতো তাহলে কেউ কম্পিউটার ব্যবহার করতো না । তাই আমাদের যদি টাইম কমপ্লেক্সিটি সম্পর্কে ভালো ধারণা থাকে, তাহলে আমরা খুব সহজে বলে দিতে পারবো যে

কোন প্রগ্রামের জন্য কোন অ্যালগরিদম সব থেকে বেশী কার্যকরী এবং কোনটায় সব থেকে কম।

এখন টাইম কমপ্লেক্সিটির একটা সহজ উদাহরণ দেখা যাক। আমরা ২ টা সংখ্যা যোগ করবো এবং তার কমপ্লেক্সিটি বের করবো :

```
number1 = 2;

number2 = 4;

sum = number1 + number2;

print(sum);
```

এখানে একটা কনস্ট্যান্ট সংখ্যক ইন্সট্রাকশন নিয়ে কাজ করে । অর্থাৎ অ্যালগরিদমের কমপ্লেক্সিটি হলো O (1) , মানে হল ইনপুটের আকার যেমনই হোক না কেন একটি কনস্ট্যান্ট টাইমে কাজ করা শেষ করবে ।

Big-O notation :

একটি অ্যালগরিদমের কর্মক্ষমতা বা কমপ্লেক্সিটি বিশ্লেষনের জন্য বিগ O নোটেশন ব্যাবহার করা হয় । অর্থাৎ বিগ O নোটেশন হলো কমপ্লেক্সিটি প্রকাশ করার মাধ্যম ।

Big-O notation কেন :

Big-O notation কে মাথায় রেখে আমরা যদি আমাদের সফটওয়্যার ব্যাবহার করি তাহলে, আমাদের সফটওয়্যার অনেক ভালো কাজ করতে পারবে। যা কিনা অনেক ফাস্টার ও হবে এবং সাইজে অতিরিক্ত যায়গাও নিবে না। আমরা নিত্যদিন যে সব অ্যাপ্লিকেশন ব্যাবহার করে থাকি তার ম্যাক্সিমাম যায়গায় অনেক অ্যালগরিদমের কাজ করছে, যার পিছনে Big-O notation ব্যাবহার করে আরো এফিশিয়েন্ট করা হয়েছে । তাই আমরা কোথাও কোনো কিছু সন্ধান করতে গেলে খুব সহজে এবং খুব দ্রুত পেয়ে যাই । তাই কার্যকর অ্যালগরিদমের তৈরীর ক্ষেত্রে Big O notation এর ধারণা থাকা অত্যন্ত প্রয়োজউদাহরণ :

O(1) :

ইনপুট যাই হোক না কেন প্রগ্রামের অপারেশন সংখ্যা ধ্রুবক বা কনস্ট্যান্ট হলে তার কমপ্লেক্সিটি সব সময় O(1) হবে অর্থাৎ ইনপুটের আকার যেমনই হোকনা কেন একটি constant টাইমে অ্যালগোরিদমটি কাজ করা শেষ করবে। যেমন :

```
num1 = 5 ;

num2 =10;

sum = num1+num2 ;

print(sum);
```

O(n):

ইনপুটের মান এর মানের উপর নির্ভর করে। n এর মান বাড়ার সাথে সাথে অপারেশনের সংখ্যা ও লিনিয়ার আকারে বাড়তে থাকে। যেমন n=8;

```
for x in range(n):

print("hello World");
```

➢ O(n²):

ইনপুটের মান এর মানের উপর নির্ভর করে। n এর মান বাড়ার সাথে সাথে অপারেশনের সংখ্যাও বর্গের সাথে সরাসরি সমানুপাতিক হারে বাড়তে থাকে। যেমন data = [2,3,4]

```
for x in data:

for y in data:

print(x, y)
```

➢ O(log n):

অ্যালগরিদম কমপ্লেক্সিটি তুলনামূলকভাবে অনেক কম। এখানে n এর জন্য কমপ্লেক্সিটি কমে log(n) গুণ হয়ে যাচ্ছে, তাই এ ধরণের অ্যালগরিদম খুবই দ্রুত এবং কম মেমোরিতেই কাজ করতে পারে।

একটি অ্যালগরিদমের সময় জটিলতা বিশ্লেষণ করার সময় আমরা তিনটি ক্ষেত্রে খুঁজে পেতে পারি যেমন :

❖ best-case :

কোনো প্রোগ্রামের থেকে খুব কম সময়ে ফলাফল খুঁজে বের করা হলো best-case। অর্থাৎ সবথেকে কম সময়ে সঠিক ফলাফল দিতে পরে সেটাই হলো best-case।

❖ average-case :

Average-case হলো ফলাফল খুঁজে পেতে অনেক সময় ও লাগবে না আবার খুব অল্প সময়েও পাবো না সেটাকে average-case বলে।

❖ worst-case

Best-case এর সম্পূর্ণ বিপরীত কেস হলো worst-case। অর্থাৎ কোনো প্রোগ্রামে ফলাফল খুঁজে পেতে বেশী সময় লাগলে সেটাকে worst-case বলে।

➢ স্পেস কমপ্লেক্সিটি :

আমরা কোন ডাটা রাখার জন্য ও কম্পিউটার ব্যাবহার করে থাকি কারন এত ডাটা মনে রাখা যায় না। এই ডাটা রাখার জন্য মেমোরি এর দরকার পরে। কিন্তু কম্পিউটারের ও কিন্তু একটা ধারন ক্ষমতা দেয়া আছে তার বেশী সে ডাটা রাখতে পারবে না। তাই আমাদের যেকোনো প্রোগ্রাম লেখার সময় অবশ্যই খেয়াল রাখেতে হবে যে সে কতটুকু মেমোরি এর জায়গা দখল

করে। আমারা যদি স্পেস কমপ্লেক্সিটি বুজি তাহলে কোন প্রগ্রামে দেখেই বলে দেয়া যাবে যে সে কতটুকু যায়গা নিবে।

যেমন : ভেরিয়েবল , অ্যারে, অবজেক্ট।

রেফারেন্সেস লিঙ্ক :

https://en.wikipedia.org/wiki/Big_O_notation

https://en.wikipedia.org/wiki/Time_complexity