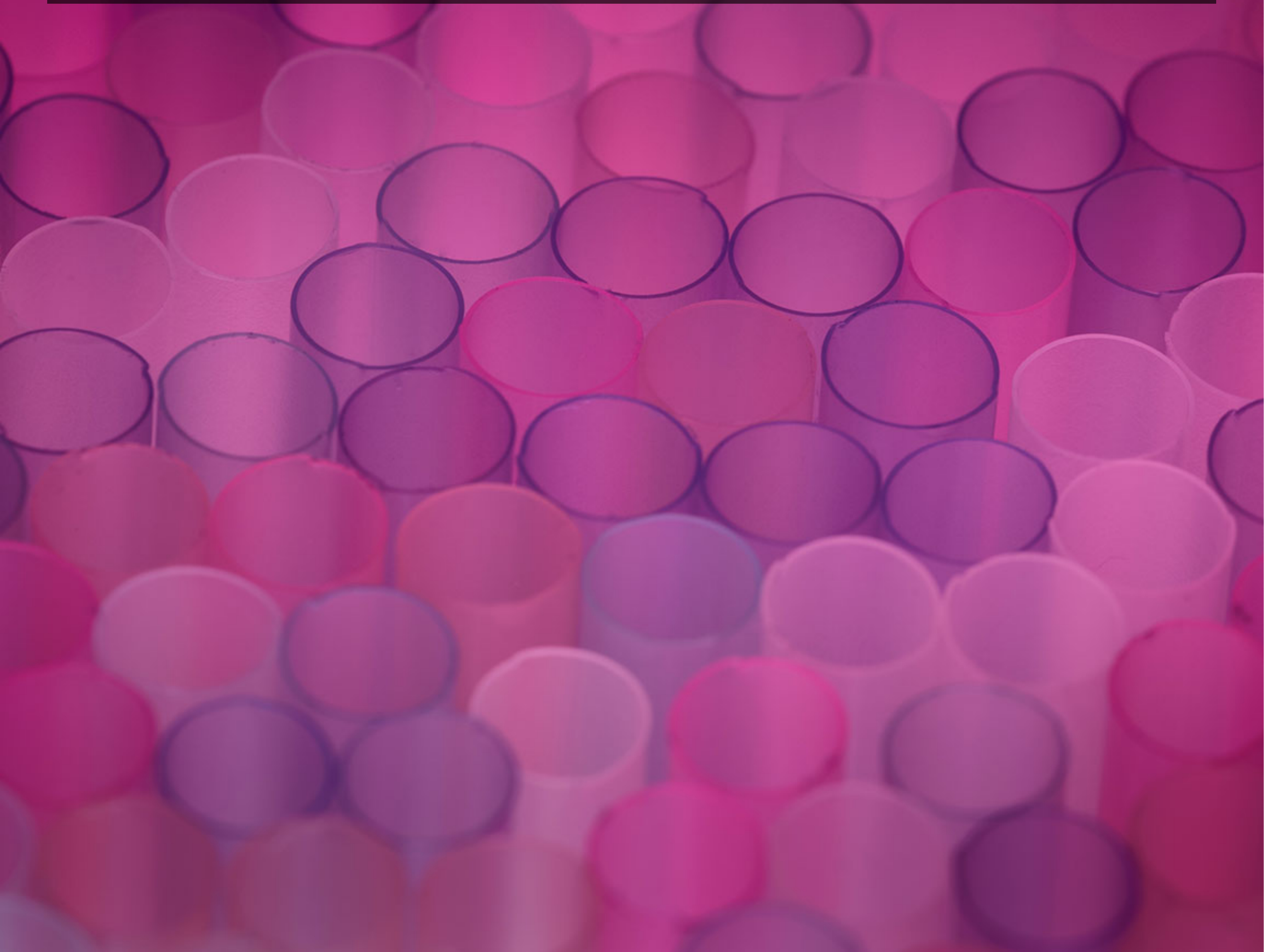
ড. মোঃ হানিফ সিদ্দিকী

ডেটা স্ট্রাকচার

প্রথম সংস্করণ



ডেটা স্ট্রাকচারের প্রাথমিক পাঠ

অবতারণা

তুমি যখন তোমার পড়ার টেবিলের বই গুছিয়ে রাখো অথবা বুকশেলফে বই সাজিয়ে রাখো, তুমি কিন্তু অবচেতনভাবেই বইগুলোকে এমনভাবে রাখো যেন কম যায়গায় অনেকগুলো বই রাখা যায় এবং পড়ার সময় সহজেই সেখান থেকে প্রয়োজনীয় বই বের করে আনা যায়। অথবা তুমি যখন হোটেলের বুফেতে খেতে যাও, প্লেটগুলোর স্ট্যাক থেকে যখন উপরের প্লেটটি নিয়ে খাবার নিতে যাও, অথবা ট্রেন বা বাসের টিকিটের জন্য লাইনে দাঁড়িয়ে অপেক্ষা কর, কিংবা কম্পিউটারের মধ্যে ফাইলগুলোকে ফোল্ডারে এবং নির্দিষ্ট ড্রাইভে সংরক্ষণ কর, তখন তোমরা কাজগুলো এজন্য কর যেন ফাইলগুলো দ্রুত খুঁজে পাও অথবা সুব্যবস্থাপনার মাধ্যমে সুচারুরূপে কাজগুলো সম্পন্ন করা যায়।

টেবিলের উপর বই যেভাবে সাজিয়ে রাখা হয়, বুফেতে থালাগুলো কিন্তু সেভাবে সাজিয়ে রাখা হয় না। অথবা বাস বা ট্রেনের টিকিট কাটা ঝামেলামুক্ত করতে মানুষের ভীড়কে ব্যবস্থাপনায় আনতে কিন্তু একইভাবে সাজানো হয় না। কম্পিউটারে ফাইলগুলো অথবা একটি বড় লাইব্রেরিতে বই সাজিয়ে রাখার জন্য কিন্তু অতিরিক্ত ব্যবস্থাপনা নিতে হয়। সমস্যার ধরণের উপর নির্ভর করে বিভিন্ন সমস্যা সমাধানের জন্য বিভিন্নভাবে বস্তুগুলোকে সাজাতে হয়।

একইভাবে, কম্পিউটারে তথ্য বা ডেটাকে এমনভাবে সাজিয়ে রাখতে হয় যেন প্রয়োজনের সময়ে সহজে বের করা যায় এবং সমস্যার সমাধান করা হয়। যেমন কম্পিউটারের ভেতর আমরা ফাইল আকারে এক একটি নির্দিষ্ট ফোল্ডার এবং ড্রাইভে তথ্য সংরক্ষণ করি। এখানেও তথ্যের বিন্যাসের কারণেই কিন্তু আমরা আমাদের কাঙ্ক্ষিত ফাইলটি পুনরায় সহজে খুঁজে পাই। তেমনি প্রতিনিয়ত আমরা Google ব্যবহার করি। কোন কিছু লিখে Google এ খুঁজি। Google একটি জটিল প্রক্রিয়ার মাধ্যমে খুব কম সময়ে আমাদের সামনে আমাদের কাঙ্ক্ষিত তথ্যভাণ্ডার এনে হাজির করে। Googleএর এই জটিল প্রক্রিয়ার মধ্যে পদ্ধতি ও টেকনোলোজি যেমন উপস্থিত একইভাবে তথ্য বা ডেটার বিন্যাস বা গঠনও সহায়ক হিসেবে কাজ করে।

একেক সমস্যা সমাধানের জন্য বাস্তবতার নিরিখেই ভিন্ন ভিন্ন কাঠামো, গঠন বা structure ব্যবহার করা হয়। তথ্যকে কম্পিউটার মেমোরিতে ভিন্ন ভিন্ন গঠনে সুসজ্জিত করে রাখা হয়। এইভাবে, বিভিন্ন রকম সমস্যার সমাধানে তথ্যকে ভিন্ন ভিন্নরূপে মেমোরিতে organize বা সজ্জিত করে রাখাকে বলা হয় ডেটা স্টাকচার। ডেটা স্ট্রাকচার তৈরির ক্ষেত্রে দুটি বিষয় মাথায় রাখতে হয়ঃ ১) ডেটা বা তথ্যের পারস্পরিক সম্পর্ক যেন বিনষ্ট না হয়, ২) যতটুকু সম্ভব এটি যেন সহজ হয় যেন দরকারের সময় effectively বা কার্যকরভাবে পাওয়া বা কাজ করা যায়।

বস্তুত effectivity বা কার্যকরতা কিভাবে বুঝবো? পৃথিবীতে সমস্যা সমধানে সকল প্রোগ্রামার সাধারণ দুটি বিষয় মাথায় রেখে অগ্রসর হন। সমস্যার সমধান বের করতে যেন ১) কম সময় লাগে এবং ২) কম মেমোরি খরচ হয়। তবে সমস্যার ধরণভেদে কখনো কখনো যেকোন একটার গুরুত্ব আরেকটা থেকে বেড়ে যেতে পারে। একটি ডেটা স্ট্রাকচার effective বা কার্যকর কিনা তা উপরিউক্ত দুটি বিষয় দিয়ে পরিমাপ করা যায়। কিন্তু কম্পিউটারের ক্ষেত্রে সময় খুব আপেক্ষিক বিষয়। এটি নিঃসন্দেহে কম্পিউটারটির কনফিগারেশন বা কম্পিউটারে উপস্থিত বিভিন্ন অংশের কর্মক্ষমতার উপরে নির্ভর করে। উচ্চ ক্ষমতা সম্পন্ন কম্পিউটারে যে কাজ সম্পন্ন করতে ১সেকেন্ড সময় লাগবে, কম ক্ষমতা সম্পন্ন কম্পিউটারে তথ্য বা ডেটা একইভাবে সজ্জিত থাকলেও বা একই ডেটা স্ট্রাকচার ব্যবহার করলেও সেই কাজ সম্পন্ন করতে ১সেকেন্ডের বেশী সময় লাগতে পারে। এমতাবস্থায়, একই সমস্যা-সেটে (problem-set) ব্যবহৃত বিভিন্ন ডেটা স্ট্রাকচারের জন্য ভিন্ন ভিন্ন সময়কে কার্যকরভাবে তুলনা করার সুযোগ থাকছে না। এমন একটা ব্যবস্থা দরকার যা একেক জায়গায় একেক রকম আচরণ করবে না। বিভিন্ন কম্পিউটারের ভিন্ন ভিন্ন কনফিগারেশন কোন প্রভাব বিস্তার করবে না। এইজন্যই বিভিন্ন ডেটা স্ট্রাকচারের কার্যকারিতা তুলনা করার জন্য একটি গাণিতিক মডেল তৈরি করা হয়েছে। এই গাণিতিক মডেলের নাম Asymptotic Notation বা অ্যাসিমটোটিক সূচক। এর মাধ্যমে ভিন্ন ভিন্ন ডেটা স্ট্রাকচারের মধ্যে বিভিন্ন বিষয়ের প্রভাবমুক্ত থেকে তুলনা করা যায়। এটি শুধুমাত্র ইনপুট সংখ্যার উপরে নির্ভর করে, কিন্তু কোনভাবেই কম্পিউটারের উপরে নির্ভরশীল নয়।

ডেটার বিন্যাস, গঠন, কাঠামো বা স্ট্রাকচার নিয়ে আলোচনা করার আগেই আমরা এই গাণিতিক ধারণাটি পরিষ্কার করে নিতে চাই যেন বিভিন্ন গঠন আলোচনার পাশাপাশি তাদের সক্ষমতার জায়গা এবং দুর্বলতা বের করতে পারি।

১। ডেটা স্ট্রাকচার কাকে বলে?
২। বাস্তব জীবনে ও কম্পিউটারে কয়েকটি ডেটা বিন্যাসের উদাহরণ দাও।
৩। ডেটা স্ট্রাকচার তৈরির সময় কি কি মনে রাখা দরকার?
৪। প্রোগ্রামাররা সবসময় কয়টি বিষয় মাথায় রাখে এবং কেন?
৫। বিভিন্ন ডেটাস্ট্রাকচারকে তুলনা করার জন্য কেন আমরা Asymptotic Notation ব্যবহার করি?

অ্যাসিমটোটিক সূচক (ASYMPTOTIC NOTATION)

অ্যাসিমটোটিক সূচক হলো এমন একটি গাণিতিক সূচক যার মাধ্যমে কম্পিউটারের তথ্যের উপরে প্রয়োগকৃত কোন অ্যালগোরিদম ইনপুট সংখ্যার পরিবর্তনের সাথে সাথে কেমনভাবে আপেক্ষিক সময়ের পরিবর্তন হতে পারে তার ধারণা দেয়। এটি প্রকৃত কোন সময়কে উপস্থাপন করে না, বরং ইনপুট সংখ্যার পরিবর্তনের সাথে সাথে তুলনামূলকভাবে সময় কতটুকু লাগতে পারে তার ধারণা দেয় মাত্র। প্রকৃত সময় উপস্থাপন না করার কারণেই এটি বেশী জনপ্রিয়, কেননা ইনপুট সংখ্যা পরিবর্তনের সাথে সাথে কম্পিউটারের গতি ও প্রযুক্তির উপর নির্ভর করে

একেক কম্পিউটারে সময়ের হেরফের একেক রকম হবে। কাজেই পৃথিবীর ভিন্ন ভিন্ন প্রান্তের কম্পিউটার ব্যবহার করে প্রাপ্ত প্রকৃত সেকেন্ড বা মিনিট দিয়ে কোন অ্যালগোরিদমের তুলনামূলক বিশ্লেষণ করা যৌক্তিক নয়। এ কারণেই কম্পিউটার অনির্ভরশীল এই অ্যাসিমটোটিক সূচক বেশী জনপ্রিয়।

যেকোন একটি মান বা বক্ররেখা যখন মুক্তভাবে অসীমে কাছাকাছি চলে আসে তাকে অ্যাসিমটোটিক বলে।

অ্যাসিমটোটিক সূচক ৩ ধরণেরঃ Big-O, Omega ও Theta notation বা সূচক।
ধরা যাক, কোন অ্যালগোরিদমে n সংখ্যক ইনপুট দেয়া হলো এবং এতে করে ঐ অ্যালগোরিদমে f(n) সংখ্যক বিভিন্ন অপারেশন সংগঠিত হলো। এই f(n) এ n চলকের উপর বিভিন্ন ঘাত মিলে একটি জটিল কোন সমীকরণ তৈরি হটে পারে, যেমনঃ $f(n) = n^2 + 2n + 1$ । নির্দিষ্ট শর্তে এই রকম বিভিন্ন ঘাতের একটি সমীকরণ থেকে কোন একটি ঘাতের সমীকরণে এনে অ্যাসিমটোটিক সূচক বের করা হয়।

পূর্ববর্তী সমীকরণ থেকে,

$$f(n) = n^2 + 2n + 1$$

$$\leq n^2 + 2n^2 + n^2;\ \ if\ n \geq\ 1$$

$$= 4n^2 = cg(n)$$

অথবা,

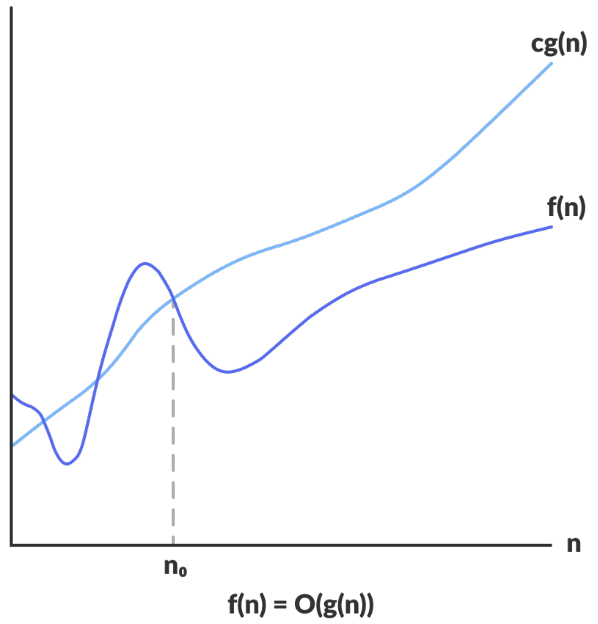
$$f(n) = n^2 + 2n + 1$$

$$= (n+1)^2$$

$$= (n+n)^2 \leq 4n^2 = cg(n);\ \ if\ n \geq 1$$

অর্থাৎ $f(n) \leq cg(n)$, যখন n এর মান 1 এর সমান বা এর চেয়ে বড়। এক্ষেত্রে n-এর মান 1 এর সমান বা এর চেয়ে বড় হলে f(n) সবসময়ই n^2এর মাত্রার চেয়ে ছোট হবে। অর্থাৎ f(n) ফাংশনটি g(n) ফাংশন দিয়ে আবদ্ধ। বিধায়, f(n) হলো O(g(n)) বা $O(n^2)$ বলা হয়। এটিই আসলে বিগ-ও (Big-O) সূচক যাকে বড় হাতের O দিয়ে প্রকাশ করা হয়। চিত্র-1.1 এ Big-O সূচক দেখানো হলো।

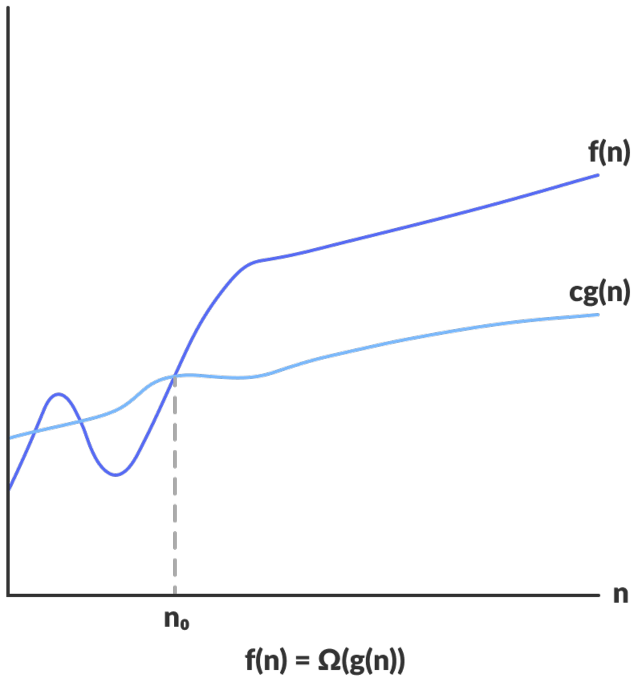
চিত্রে n_0হলো একটি সর্বনিম্ন মান যার চেয়ে বড় সকল n এর জন্য f(n) এর মান cg(n) এর চেয়ে ছোট হবে, অর্থাৎ f(n) n_0 এর চেয়ে বড় যেকোন মানের জন্য cg(n) দ্বারা আবদ্ধ, যেখানে c একটি ধনাত্মক constant বা ধনাত্মক ধ্রুবক। কাজেই, $f(n) = O(g(n))$.



চিত্র-1.1 বিগ-ও সূচক *f(n)*=O(*g*(*n*))

n_0 এর চেয়ে বড় যেকোন সংখ্যক ইনপুটের জন্য কোন অ্যালগোরিদমের রানিং সময় বিগ-ও বা $O(g(n))$ এর চেয়ে কম হবে না।

একইভাবে, যদি কোন একটি n_0 থেকে বড় সকল n এর জন্য f(n) এর মান সবসময় cg(n) এর চেয়ে বড় হয় অর্থাৎ, cg(n) ফাংশনটি f(n) দ্বারা আবদ্ধ হয় বা cg(n) ফাংশনটি f(n) এর নীচের সীমা নির্দেশ করে, যেখানে c একটি ধনাত্মক constant বা ধনাত্মক ধ্রুবক। কাজেই, $f(n) = \Omega(g(n))$ অর্থাৎ ওমেগা সূচক বলে। চিত্র-1.2 এ ওমেগা সূচক দেখানো হলো।

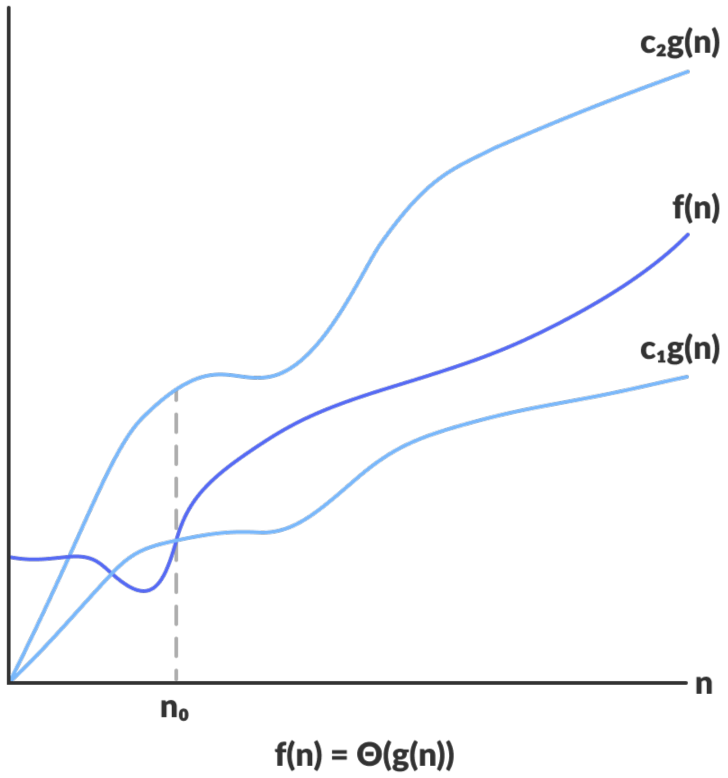


চিত্র ১.২ বিগ-ওমেগা সূচক

চিত্রে দেখা যায় যে, n_0 এর চেয়ে বড় যেকোন সংখ্যক ইনপুটের জন্য কোন অ্যালগোরিদমের কমপক্ষে $\Omega(g(n))$ রানিং সময় প্রয়োজন হবে।

অপরদিকে, থিটা সূচক বা Theta notation এর ক্ষেত্রে কোন একটি ফাংশন শুধুমাত্র ধ্রুবকের পরিবর্তনের মাধ্যমে সর্বোচ্চ এবং সর্বনিম্ন সীমা তৈরি করতে পারে। চিত্র-1.3 এ দেখা যায় যে, n_0 এর চেয়ে বড় যেকোন সংখ্যক ইনপুটের জন্য শুধুমাত্র দুটি ধনাত্মক দ্রুবরাশি c_1 ও c_2 কারণে f(n) দুটি ফাংশন $c_1g(n)$ ও $c_2g(n)$ দ্বারা আঁটভাবে এমনভাবে আবদ্ধ যেন $0 \leq c_1g(n) \leq f(n) \leq c_2g(n)$ হয়। অর্থাৎ দুটি ধনাত্মক দ্রুবরাশি c_1 ও c_2 এর মাধ্যমে g(n) ফাংশনটি f(n) ফাংশনকে আঁট করে আবদ্ধ করতে পারে। কাজেই, $f(n) = \Theta(g(n))$ যা চিত্র-1.3তে দেখানো হয়েছে।

থিটা সূচক কোন ফাংশনকে আঁট করে আবদ্ধ করে বলে এটি বের করা অপেক্ষাকৃত জটিল মনে হতে পারে। কিন্তু



চিত্র ১.৩ বিগ-থিটা সূচক

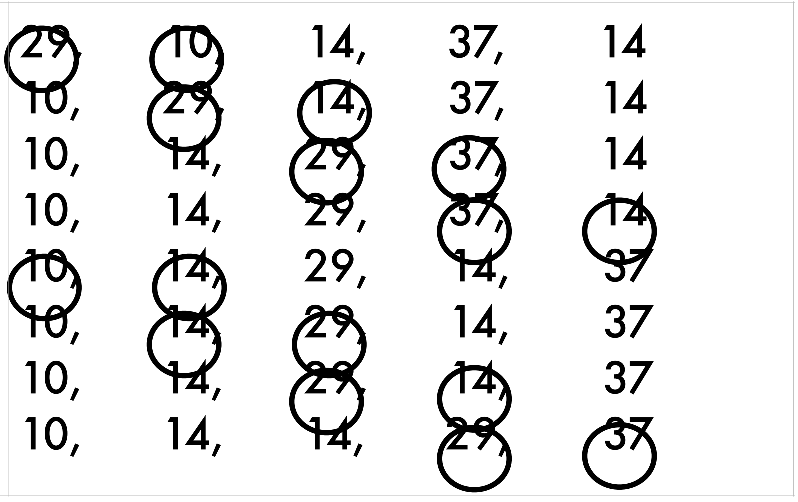
বস্তুত কোন ফাংশনকে যদি f(n) দ্বারা প্রকাশ করা হয়, তাহলে f(n) এ n-এর যে সর্বোচ্চ ঘাত রয়েছে, নির্দ্বিধায় তাঁকে থিটা সূচক দিয়ে প্রকাশ করা যায়। যেমন, $f(n) = n^2 + 2n + 1$ হলে $f(n) = \Theta(n^2)$ অথবা $f(n) = 3n^3 - 2n^2 + n - 2$ হলে নিঃসন্দেহে $f(n) = \Theta(n^3)$ পাওয়া যায়।

১। Asymptotic Notation কাকে বলে?
২। বিগ-ও, বিগ-ওমেগা এবং বিগ-থিটা সূচক সচিত্র বুঝিয়ে দাও।
৩। বিগ-থিটা সূচকে আঁট-আবদ্ধ বলা হয়? একটি ফাংশন থেকে কিভাবে থিটা সূচক বের করা যায়?

বিগ-ও, বিগ-ওমেগা ও বিগ-থিটা এই তিনটি অ্যাসিম্পটোটিক সূচকের মাধ্যমে আমরা সকল অ্যালগোরিদম ও ডেটা স্ট্রাকচারের রানিং সময় তুলনা করতে পারি এবং কোনটি দ্রুতগতির তা নির্ধারণ করতে পারি। কিন্তু সমস্যা হলো, একই অ্যালগোরিদম ও ডেটা স্ট্রাকচার ভিন্ন পরিস্থিতিতে (case) ভিন্ন ভিন্ন অ্যাসিম্পটোটিক সূচক নির্দেশ করতে পারে। এই পরিস্থিতি(case)গুলোকে ৩টি ভাগে ভাগ করা যায়ঃ সর্বোত্তম পরিস্থিতি (best case), সবচেয়ে খারাপ পরিস্থিতি (worst case), এবং গড় পরিস্থিতি (average case)। একটি অনুক্রম (sorting) উদাহরণের মাধ্যমে অ্যাসিম্পটোটিক সূচকসহ বিভিন্ন পরিস্থিতি আলোচনা করা যাক। ধরা যাক, ৫টি নম্বর আছে 29, 10, 14, 37, 14. এগুলোকে অনুক্রম অর্থাৎ ছোট থেকে বড় ক্রমে সাজাতে হবে। পানির নীচ থেকে যেমন বুদবুদ (Bubble) তৈরি হয়ে সবচেয়ে বড় বুদবুদ সবার আগে ধীরে ধীরে উপরের দিকে উঠতে থাকে, প্রথম থেকে শুরু করে এই নম্বর-ধারার সবচেয়ে বড় নম্বর ধীরে ধীরে সবার শেষে স্থির হয়।

এই ধারার প্রথম নম্বর 29 বলে 29 থেকে শুরু করে ধীরে ধীরে পরের নম্বর 10এর সাথে তুলনা করলে 29 বড় হয় (চিত্র-1.4এর ডানে 1নং লাইন) । কাজেই 29 ও 10 স্থান পরিবর্তন করবে যেন 29 শেষের দিকে ধাবিত হতে পারে এবং তারপর 14এর সাথে তুলনা করবে (চিত্র-1.4এর ডানে 2নং লাইন)। এবারও 29 বড় হওয়ায়, 29 ও 14 যায়গা বদল করবে এবং পরের নম্বর 37এর সাথে তুলনা করবে (চিত্র-1.4এর ডানে 3নং লাইন)। এবার 29 ও 37এর মধ্যে কিন্তু 37 বড় হবার কারণে জায়গা পরিবর্তন না করে 37এর সাথে পরের নম্বর 14 তুলনা করা হবে (চিত্র-1.4এর ডানে 4নং লাইন) এবং 37 বড় হবার কারণে জায়গা বদল করবে। যেহেতু একবার হলেও নম্বরগুলোর মধ্যে জায়গা পরিবর্তন ঘটেছে, কাজেই আবার প্রথম থেকে একই ধারাবাহিকতা শুরু করবে। যেমন প্রথম নম্বর 10কে দ্বিতীয় নম্বর 14এর সাথে তুলনা করে দ্বিতীয় নম্বর বড় হবার কারণে জায়গা বদল হবে না এবং 14এর সাথে পরবর্তী নম্বর 29 এর তুলনা করা হবে। পুনরায় জায়গা বদল না করে পরের নম্বর 14 এর সাথে 29 এর তুলনা করা হবে। 29 নম্বরটি 14 এর চেয়ে বড় হবার কারণে জায়গা বদল করবে এবং 29এর সাথে 37 তুলনা করবে এবং জায়গা বদল হবে না। এবারও জায়গা বদল হবার কারণে প্রথম থেকে শুরু হবে। কিন্তু এবার কোন জায়গায় বদল হবে না বলে লুপ বন্ধ হয়ে যাবে এবং নিশ্চিত হওয়া যাবে যে ধারাটি অনুক্রম হয়ে গেছে। এই অ্যালগোরিদমকে Bubble Sort বলা হয়।

```
do
  swapped = false
  for i = 1 to n-1
    if leftElement > rightElement
      swap(leftElement, rightElement)
      swapped = true
while swapped
```



চিত্র-1.4 বাবল-সর্ট অ্যালগোরিদমের Pseudo-Code যা প্রোগ্রামিং ভাষায় নয় বরং প্রাকৃতিক ভাষায় লিখিত

Bubble sortএর ক্ষেত্রে 10, 14, 14, 29, 37 ধারাটি খেয়াল করি। এই ধারার উপরে Bubble sort প্রয়োগ করলে 10 এর সাথে 14, 14 এর সাথে 14, 14 এর সাথে 29, 29 এর সাথে 37 তুলনা করে কোন জায়গা বদল হবে না বিধায় প্রথম বারেই লুপ বন্ধ হবে। কাজেই এক্ষেত্রে Asymptotic Notation হলো O(n) যা সর্টিং এর ক্ষেত্রে সবচেয়ে ভাল অবস্থা।

কিন্তু ধারাটি যদি 37, 29, 14, 14, 10 হতো, তাহলে প্রতিবারই জায়গা বদল হতো। আউটার লুপ n বার এবং প্রতিবারই ইনার লুপ (n-1) রান হবে। কাজেই f(n) হতো n(n-1) অর্থাৎ এই ক্ষেত্রে কমপ্লেক্সিটি হতো $O(n^2)$.

f(n) = n(n-1)

< n(n)

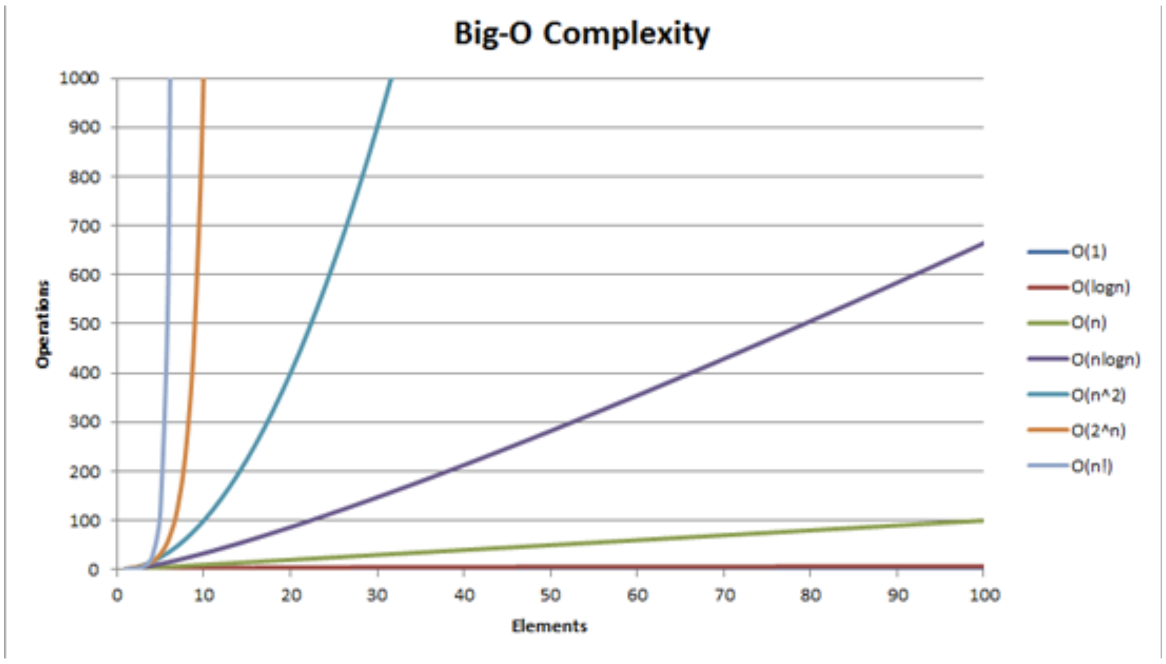
$= n^2$

আবার ধারাটি যদি প্রথমেই আলোচিত ধারার মত না অনুক্রম বা বিপরীতক্রম অনুযায়ী আছে তাহলে গড়ে n/2 বার Bubble ঘটবে বা লুপ চলবে। তাহলে হতো $f(n) = \frac{n}{2}(n-1)$, অর্থাৎ এই ক্ষেত্রেও কমপ্লেক্সিটি হতো $O(n^2)$.

$$f(n) = \frac{1}{2}n(n-1)$$

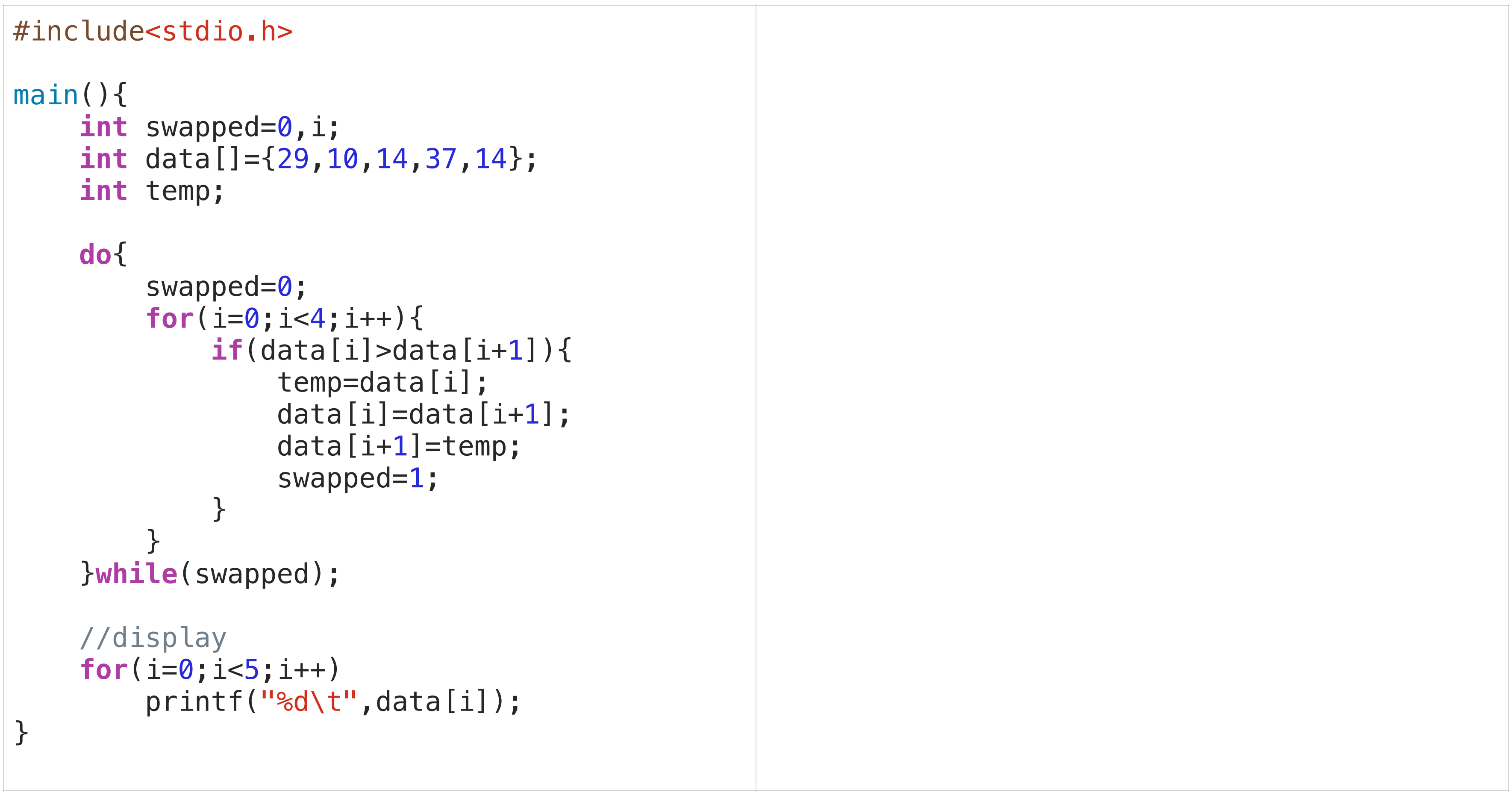
দেখা যাচ্ছে অ্যালগোরিদম একই থাকা সত্ত্বেও তথ্যের বিভিন্ন অবস্থার কারণে সর্বোত্তম, সবচেয়ে খারাপ বা গড় পরিস্থিতির সৃষ্টি হয়। যখন আগে থেকেই নম্বরের ধারাটি অনুক্রম থাকে তখন কমপ্লেক্সিটি সবচেয়ে ভাল O(n) থাকায় এই অবস্থাকে সর্বোত্তম পরিস্থিতি (best case) বলে। আবার যখন নম্বরের ধারাটি বিপরীত ক্রমে থাকে তখন কমপ্লেক্সিটি সবচেয়ে খারাপ $O(n^2)$ থাকায় এই অবস্থাকে সবচেয়ে খারাপ পরিস্থিতি (worst case) বলে। অনুক্রম বা বিপরীতক্রম নয়, বরং যেকোন অনিয়মিত ধারায় বিদ্যমান নম্বরগুলোর ক্ষেত্রেও কমপ্লেক্সিটি $O(n^2)$ হয় যাকে গড় পরিস্থিতি বা average case বলা হয়।

চিত্র-1.5 এ n বৃদ্ধির সাথে সাথে কোন কোন অর্ডার বা কমপ্লেক্সিটি কি হারে বৃদ্ধি পায় তার একটি তুমনামূলক চিত্র দেয়া হলো। চিত্রে দেখা যায় O(1) অর্থাৎ এই ক্ষেত্রে f(n)=c যেখানে c একটি ধ্রুবক। এছাড়া O(log n) এর বৃদ্ধির হার সবচেয়ে কম। তারপর n সাথে linearly বা সরলহারে বৃদ্ধি পায় O(n)। n এর সাথে log n গুণ থাকার কারণে এটি সরলহারের চেয়ে সামান্য বেশী হারে বাড়লেও O(n log n)এর মাত্রা অত্যন্ত সহনীয়। কিন্তু, $O(n^2)$ সঙ্গতকারুণেই লাফিয়ে লাফিয়ে এক্সপোনেন্টহারে বা দ্রুততার সাথে বৃদ্ধি পেতে থাকে। এর থেকেও $O(2^n)$ বা O(n!) এর বৃদ্ধির হার অত্যধিক বেশী। $O(2^n)$ বা O(n!) খুব দ্রুতই অসীমের দিকে রওনা হয় বলে বেশী ইনপুট উপাদান নিয়ে কাজ করার সুযোগ থাকে না। এই বৃদ্ধির হার সম্পর্কে সম্যক ধারণা থাকলে কোন অ্যালগোরিদম কতটুকু ভাল তা ব্যাখ্যা করা সহজ হয়ে যায়।



চিত্র-1.5 উপাদান সংখ্যা n এর সাথে বিভিন্ন কমপ্লেক্সিটির বৃদ্ধির হার

চিত্র-1.4 এ প্রোগ্রামিং ভাষার উপর অনির্ভরশীল প্রাকৃতিক ভাষায় (সাধারণ মানুষ বুঝতে পারে এমন ভাষা) লিখিত pseudo-code কে প্রোগ্রামিং ভাষা C তে লিখলে চিত্র-1.6 এ লিখিত কোড হয়।



```
#include<stdio.h>

main(){
    int swapped=0,i;
    int data[]={29,10,14,37,14};
    int temp;

    do{
        swapped=0;
        for(i=0;i<4;i++){
            if(data[i]>data[i+1]){
                temp=data[i];
                data[i]=data[i+1];
                data[i+1]=temp;
                swapped=1;
            }
        }
    }while(swapped);

    //display
    for(i=0;i<5;i++)
        printf("%d\t",data[i]);
}
```

চিত্র-1.6 C-তে লিখিত bubble sort অ্যালগোরিদম

C তে লিখিত কোডে অ্যারে ব্যবহৃত হয়েছে। ধরে নেয়া যাক, data[i] হলো leftElement, তাহলে এর পরের নম্বরটি (অর্থাৎ data[i+1]) rightElement হবে।

এবার একটি সমস্যা নিয়ে ভাবা যাক। ধরে নাও তোমার কাছে দুটি তরলভর্তি পাত্র আছে। একপাত্রে পানি আছে। আরেকপাত্রে দুধ। এবার, দুধের পাত্রে পানি আর পানির পাত্রে দুধ swap বা অদলবদল করবে কিভাবে? নিশ্চয় তুমি তৃতীয় আরেকটি পাত্রের সাহায্য নেবে, তাই না? কাজেই leftElement বা data[i]-কে rightElement বা data[i+1] তে নিয়ে যাবার জন্য temp নামে একটি ধারকের সাহায্য নেয়া যাক। data[i] কে temp এ নিয়ে data[i+1]-কে data[i]-তে সহজেই নেয়া যাবে। তারপর temp-কে data[i+1]-তে নিলেই অদলবদল হয়ে যাবে। অতএব pseudo-codeটি প্রোগ্রামিং ভাষা C তে লেখা যায় নিচের মত করে।

১। Asymptotic Notation এর ক্ষেত্রে পরিস্থিতি (case) কত প্রকার ও কি কি?

২। বাবল সর্ট (Bubble sort) অ্যালগোরিদমটি লেখ।

৩। যেকোন একটি নম্বরের ধারার উপরে বাবল সর্ট অ্যালগোরিদমটি ব্যাখ্যা কর।

৪। বাবল সর্ট অ্যালগোরিদম ব্যবহার করলে 1 থেকে 1-মিলিয়ন পর্যন্ত অনুক্রমে থাকা নম্বরগুলোর ক্ষেত্রে কতবার অপারেশন করতে হবে?

৫। বাবল সর্ট অ্যালগোরিদম ব্যবহার করলে 1-মিলিয়ন থেকে 1 পর্যন্ত বিপরীতক্রমে থাকা নম্বরগুলোর ক্ষেত্রে কতবার অপারেশন করতে হবে?

৬। বাবল সর্টের ক্ষেত্রে সর্বোত্তম পরিস্থিতি(best case) ও সবচেয়ে খারাপ পরিস্থিতি (worst case) ব্যাখ্যা কর।

৭। নিচের ফাংশনগুলোর complexity বের কর এবং সর্বোচ্চ কত ইনপুটের জন্য এটি কাজ করতে সক্ষমঃ

ক) $f(n) = n^2 + 3n + 112$

খ) $f(n) = n^3 + 999n + 112$

গ) $f(n) = 6 \times log\ n + n \times log\ n$

ঘ) $f(n) = 2^n + n^2 + 100$

ঙ) $f(n) = n + (n-1) + (n-2) + ... + 1$