আসিফের হ-য-ব-র-ল

জানি জানার শেষ নাই, তবু শুরু করতে দোষ কোথায়??

কনটেস্ট টাইমে Algorithm Selection

Published on **November 12, 2013** | Time **18:00**

হুম, তুমি এখন কনটেস্টে আছো, হাতে প্রবলেম সেটটা পেয়েছো। প্রবলেমগুলার মধ্যে দিয়ে যাওয়ার সময় পেয়ে গেলে একটা পরিচিত প্রবলেম, তোমাদের মনে হচ্ছে এটা সল্ভ করতে পারবে। প্রবলেম পড়ার পরপরই তুমি এবং তোমার জিনিয়াস টীমমেট দুজন বাঘাবাঘা অ্যালগরিদম দাড় করায়ে ফেললে, তার মধ্যে থেকে সব থেকে সিম্পলটা সিলেক্ট করে কোড করার ট্রাই করতেছো। But there is a catch! তোমার অ্যালগরিদম কি আসলেই efficient??

ভালো কনটেস্ট প্রোগ্রামাররা এমন অ্যালগরিদম সিলেক্টশন বা বানানোর সময় প্রবলেমে দেওয়া ইনপুট দেখে অ্যালগরিদমটার কমপ্লেক্সিটিটা হিসাব করে নেয়। তার যথেষ্ট ভালো কারনও আছে। সাধারনত আমরা কনটেস্ট প্রোগ্রামাররা ধরে নি এখনকার কম্পিউটারগুলা 10^7 টি instructions মোটামুটি ১ সেকেন্ডে execute করতে পারে। এখন তুমি কনটেস্টে একটা অ্যালগরিদম তৈরী করে ফেললে যেটা কিনা ইনপুট সাইজের সাথে বিবেচনা করলে 10^8 বা তার বেশি হয়ে যাচ্ছে। কোড সাবমিট করলে জাজ তো তোমাকে TLE verdict দিতে বাধ্য হবেনই। তাই কোন অ্যালগরিদম চিন্তা করার আগে অবশ্যই মাথায় রাখতে হবে worst case scenario এর ব্যাপারটা। প্রবলেমে বলা থাকে সাধারনত যে testcases কতগুলা হবে এবং maximum input size কতগুলা থাকে। Worst case complexity = maximum test cases * maximum input size হবে (যদিও এই নিয়মেই সব প্রবলেমের কমপ্লেক্সিটি হিসাব করতে যেয়ো না, সমস্যার বর্ণনায় কিছু তথ্য এক্সট্রা দেয়া থাকতে পারে)।

তো তুমি যখন তোমার worst case complexity পেয়ে গেলে তাইলে এখন খুব সহজেই টীমমেটদের সাথে গল্প করে most efficient এবং most simpler অ্যালগরিদমটি সিলেক্ট করতে পারবে। কমপ্লেক্সিটির কথা বিবেচনা না করে যদি কোড করা শুরু করো তাইলে দেখবে কোড-টোড করা শেষ, কিন্তু TLE খাচ্ছো বারংবার।

ইনপুট সাইজ এবং টেস্টকেইজ দেখে কনটেস্টের সময়ই কিছু সাম্ভাব্য অ্যালগরিদম guess করতে পারবে। যদিও সব সময় এমনটা হবে এমন কোনো গ্যারান্টি নাই। ইনপুট সাইজ দেখে সব থেকে খারাপ complexity এর সেসব অ্যালগরিদম ব্যবহার করতে পারবে (অথচ যেগুলা ব্যবহার করলেও TLE খাবে না) সেটার একটা টেবিল দিলাম। চেষ্টা করবে যেন ইনপুট সাইজের সাথে এর সমতুল্য বা তার চে আরও efficient অ্যালগরিদম খুজে বের করে তারপর প্রবলেম সলভ করার।

Maximum N	**Maximum complexity can be applied**	**Algorithms**	**Possible data structures to use**
1,000,000,000 and higher	log (n), sqrt (n)	binary search, ternary search, fast exponentiation, euclid algorithm	
10,000,000	n, n (log (log (n))), n log(n)	set intersection, Eratosthenes sieve, radix sort, KMP, topological sort, Euler tour, strongly connected components, 2sat	disjoint sets, tries, hash_map, rolling hash deque
1,000,000	n log n	sorting, divide and conquer, sweep line, Kruskal, Dijkstra	segment trees, range trees, heaps, treaps, binary indexed trees, suffix arrays
100,000	$n \log^2 n$	divide and conquer	2d range trees
50,000	$n^{1.585}$, n sqrt n	Karatsuba, square root trick	two level tree
1000 - 10,000	n^2	largest empty rectangle, Dijkstra, Prim (on dense graphs)	
300-500	n^3	all pairs shortest paths, largest sum submatrix, naive matrix multiplication, matrix chain multiplication, gaussian elimination, network flow	
30-50	n^4, n^5, n^6		
25 - 40	$3^{n/2}$, $2^{n/2}$	meet in the middle	hash tables (for set intersection)
15 - 24	2^n	subset enumeration, brute force, dynamic programming with exponential states	
15 - 20	$n^2 2^n$	dynamic programming with exponential states	bitsets, hash_map
13-17	3^n	dynamic programming with exponential states	hash_map (to store the states)
11	n!	brute force, backtracking, next_permutation	
8	n^n	brute force, cartesian product	

Keep coding... :)

ছোট্ট করে আমার সম্পর্কে...

আমি (আবু আসিফ খান চৌধুরী), বর্তমানে Secure Link Services | SELISE Rockin' Software এ Software Engineer হিসেবে কাজ করতেছি। পড়াশুনা করেছি Rajshahi University of Engineering and Technology থেকে। প্রোগ্রামিং ও অ্যালগরিদমের প্রতি মারাত্মক আকর্ষন এবং নতুন কিছু শেয়ার করার জন্য এখানে লেখালিখি করি এবং করবো। ব্লগটাতে আমি কনটেস্ট প্রোগ্রামিং, অ্যালগরিদম, এবং এ বিষয়ক আমার জানা তথ্যগুলা সহজ ভাষায় লেখার চেষ্টা করবো।

আমাকে ফলো করতে পারেন
গিটহাব গুগল+ লিঙ্কডইন

মোট পেজভিউ

036196

Back to top