ডায়নামিক প্রোগ্রামিং ভাল করে শেখার জন্যে প্র্যাকটিসের কোন বিকল্প নেই। আর প্র্যাকটিসের জন্যে বেশিরভাগ সময় LightOj এর প্রব্লেমগুলো রিকমেন্ড করা হয়। মোট ১০৬টা ডিপি প্রব্লেম আছে এই অনলাইন জাজে এবং প্রত্যেকটাই তোমাকে নতুন নতুন কিছু শিখতে সাহায্য করবে। প্রব্লেমগুলো একে একে করতে থাকলেই দেখবে, তোমার ডিপি প্রব্লেম সলভ করার স্কিল আস্তে আস্তে বাড়ছে। কিন্তু লাইটওজের প্রব্লেমগুলো সলভের সময় একটা বড় সমস্যা দেখা দেয়, সেটা হল, কোন প্রব্লেমের সল্যুশন আইডিয়া মাথায় না আসলে অকূল পাথারে পড়তে হয়, কারণ বেশিরভাগ প্রব্লেমেরই কোন ভাল এডিটোরিয়াল নেই। ফোরামে গুতাগুতি করে হয়ত কোন হিন্টস পাওয়া যায়, নাহলে শেষমেশ অনলাইনে থাকা সল্যুশন কোডের আশ্রয় নিতে হয়। আর প্রব্লেম সলভিংয়ে কোন আইডিয়া না নিয়েই অন্যের কোড দেখা দণ্ডনীয় অপরাধ!

এই সমস্যার কথা মাথায় রেখেই লাইটওজের ডিপি প্রব্লেমগুলো নিয়ে আলোচনা করার চিন্তা মাথায় এসেছে। তবে এই পর্বগুলো পড়ার আগে রিকার্শন, ডায়নামিক প্রোগ্রামিং, ওভারল্যাপিং সাবপ্রব্লেমস, বেসকেস, স্টেট সম্পর্কে কিছু ধারণা থাকতে হবে। কারণ এটা ডায়নামিক প্রোগ্রামিং এর টিউটোরিয়াল না। তাই বেসিক জিনিস নিয়ে আলোচনা করা হবে না।

বেশ কিছু পর্বে ধীরে ধীরে আমরা প্রব্লেমগুলো নিয়ে আলোচনা করার চেষ্টা করব। বুঝতেই পারছো, সরাসরি কোন কোড এখানে দেয়া হবে না! আর কিছু স্টেপ অনুসরণ করে এই পর্বগুলো পড়া ভালঃ

১) প্রথমেই প্রব্লেম স্টেটমেন্ট ভাল মত পড়া এবং স্যাম্পল টেস্টকেসগুলো বুঝতে চেষ্টা করা। খুব সংক্ষিপ্ত আকারে স্টেটমেন্টের বর্ণনা দেয়া হবে, কিন্তু সেটা পড়ার আগে অবশ্যই মূল প্রব্লেম পড়ে নিতে হবে।

২) প্রব্লেমের শুরুতেই কিছু হিন্টস দেয়া থাকবে। হিন্টসগুলো দেখে অন্তত নির্দিষ্ট কিছু সময় প্রব্লেমটা নিয়ে ভাবা (হতে পারে ১-২ ঘন্টা কিংবা তারও বেশি সময়)। এখানে উল্লেখ্য যে, হিন্টসে অনেকসময় ক্ল্যাসিক কিছু ডিপি প্রব্লেম (যেমন LCS, LIS, Coin Change) অথবা কিছু পরিচিত সলভিং টেকনিক(যেমন Bitmask Dp, Digit Dp) এর নাম দেয়া থাকতে পারে, সেক্ষেত্রে অবশ্যই সেই টপিকটা সম্পর্কে কিছু ধারণা থাকতে হবে। যেহেতু এসব টপিকের উপর প্রচুর টিউটোরিয়াল অনলাইনে পাওয়া যায়, তাই এগুলো নিয়ে বিস্তারিত আলোচনা এখানে করা হবে না।

৩) যখন মনে হবে, হিন্টস দেখেও তোমার পক্ষে কোনভাবেই এই প্রব্লেম সলভ করা সম্ভব হচ্ছে না, ঠিক তখনই সল্যুশন আইডিয়া পড়া শুরু করবে। এক্ষেত্রেও অনেক সময় সল্যুশন আইডিয়া বিভিন্ন পার্টে লেখা থাকতে পারে। সেক্ষেত্রে প্রতি পার্টের পরে কিছু সময় ধরে সল্যুশন খোঁজার চেষ্টা করবে, তারপর পরের পার্ট পড়তে শুরু করবে।

৪) কোড `Accepted` না হলে প্রথমে ফোরাম, তারপর `uDebug` এর টেস্টকেসগুলো দিয়ে চেক করবে। তারপরেও বাগ খুঁজে বের করতে না পারলে সেটা এখানে অথবা ফেসবুক পেজের কমেন্টে দিতে পারো। অবশ্যই `ideone/pastebin` এ কোড কপিপেস্ট করে সেটার লিঙ্ক দিতে হবে। সরাসরি কোড কমেন্টে দিলে সেটা ডিলিট করে দেয়া হবে।

শুরুতেই কঠিন কঠিন নিয়মকানুন দেখে ভড়কে যাবার কারণ নেই! 😀 এটা শুধুই একটা সাজেশন, যেভাবে পড়লে সেটা তোমার কাছে সবচেয়ে বেশি ইফেক্টিভ হবে, সেভাবেই পড়বে! এবার তাহলে শুরু করা যাক!

1004 – Monkey Banana Problem

প্রব্লেম ডিস্ক্রিপশনঃ `2*N - 1 row` এর একটা অ্যারে দেয়া আছে। অ্যারেটা দেখতে কেমন হবে, সেটা ছবিতেই দেখানো হয়েছে। প্রতিটা ঘরে কিছু ভ্যালু আছে। একদম উপরের রো থেকে শুরু করে নিচের রো তে পৌছাতে হবে। যেই ঘরে যাবে, সেই ঘরের ভ্যালু যোগ হবে। কোন রো-এর একটা ঘর থেকে তার পরের রো এর এডজাসেন্ট ঘরগুলোতে যাওয়া যায়। টার্গেট হচ্ছে, ভ্যালুগুলোর যোগফল ম্যাক্সিমাইজ করা।

হিন্টসঃ রক ক্লাইম্বিং

সল্যুশন আইডিয়াঃ খুবই সহজ একটা প্রব্লেম। কোন রো এর কোন ঘর থেকে পরের রো এর কোন কোন ঘরে যাওয়া যায়, সেটা ঠিক করে ফেললেই তোমার কাজ শেষ।

ডিপি স্টেট হবে `(r,c)` অর্থাৎ তুমি কোন ঘরে (কোন রো এর কোন কলামে) আছো। একদম শেষ রো-তে পৌঁছে গেলে আর কোথাও যাওয়া যাবে না, তাই এটাই বেসকেস। ডিপি রিকারেন্সটা সংক্ষেপে লেখা যায় এভাবেঃ

$$dp(r,c) = val[r][c] + max(allpossibledp(nxtr, nxtc));$$

এখানে `nxtr, nxtc` হচ্ছে কোন স্টেট `(r,c)` থেকে পরবর্তীতে যতগুলো স্টেটে যাওয়া যায়।

1005 - Rooks

প্রব্লেম ডিস্ক্রিপশনঃ একটা `n*n` সাইজের দাবার বোর্ডে `k` সংখ্যক নৌকার গুটিকে কতভাবে বসানো যাবে যাতে কোন নৌকার গুটি অন্য কোন নৌকার গুটিকে খেতে না পারে। নৌকা তার নিজের রো এবং নিজের কলামে থাকা সব গুটিকে খেতে পারে।

হিন্টসঃ কম্বিনেটরিক্স

সল্যুশন আইডিয়াঃ প্রতি রো-তে অথবা প্রতি কলামে একটাই নৌকার গুটি বসানো যাবে, কখনোই এর বেশি বসানো সম্ভব না। `n` সংখ্যক রো-তে `k` সংখ্যক গুটি বসানো যায় কতভাবে? হ্যা, ঠিক ভেবেছো, ${}^{n}C_{k}$ সংখ্যক ভাবে।

প্রতি রো-তে যেই গুটিগুলো বসিয়েছি, সেগুলো কিন্তু আবার নিজ রো-তেই বিভিন্ন কলামে বিন্যস্ত হতে পারে। প্রথম রো এর গুটি `n` সংখ্যক কলামের মধ্যে যেকোন একজায়গায় বসতে পারে। দ্বিতীয় রো এর গুটির জন্যে একটি কলাম কিন্তু কমে গেল! তার জন্যে বাকি থাকল `(n-1)` সংখ্যক কলাম। এভাবে `k` তম রো এর গুটির জন্যে বাকি থাকবে `(n-k+1)` সংখ্যক কলাম। কম্বিনেটরিক্সের গুণন বিধি অনুযায়ী সবগুলো উত্তর গুণ হবে।

এটা মূলত কম্বিনেটরিক্স প্রব্লেম। এটাকে ডিপিতে রাখা হয়েছে কারণ ${}^{n}C_{k}$ এর রিকার্সিভ সূত্র আছে, যা ডিপির মাধ্যমে বের করা যায়।

তাহলে সল্যুশনটা হবে এরকমঃ

$$ans = {}^{n}C_{k} * n * (n-1) * (n-2) * \dots\dots * (n-k+1)$$

1011 – Marriage Ceremonies

প্রব্লেম ডিস্ক্রিপশনঃ N pair সংখ্যক ছেলে ও মেয়ের মধ্যে বিয়ে দিতে হবে। কোন ছেলে ‘i’ এর সাথে কোন মেয়ে ‘j’ এর বিয়ে দিলে একটা নির্দিষ্ট ভ্যালু যোগ হবে। টার্গেট হচ্ছে, এমনভাবে বিয়েগুলো সংঘটিত করা যাতে এই ভ্যালু ম্যাক্সিমাম হয়। খেয়াল রেখো, কোন ছেলের সাথে একাধিক মেয়ে কিংবা কোন মেয়ের সাথে একাধিক ছেলের বিয়ে দেয়া যাবে না।

হিন্টসঃ

প্রথমে কন্সট্রেইন্টস দেখো। N এর ভ্যালু দেখে কোনকিছু আন্দাজ করতে পারো? (উত্তর দেখতে ক্লিক করো)

সল্যুশন আইডিয়াঃ এটা একদম বেসিক বিটমাস্ক ডিপি। ডিপি স্টেট হবে (boy,mask). মাস্কে থাকবে তুমি কোন কোন মেয়ের সাথে বিয়ে দিয়েছো সেটা। প্রথমে মাস্ক অবশ্যই 0 হবে, কারণ কোন মেয়ের সাথেই প্রথমে বিয়ে ঠিক করা হয়নি। প্রতিটা ছেলের জন্যে যে যে পাত্রী বাকি আছে(এদের বিট অফ থাকবে, অর্থাৎ ০ থাকবে), তাদের সাথে বিয়ে দেয়ার চেষ্টা করবে। মাস্কে সেই পাত্রীর ইন্ডেক্স সেট করে পরবর্তী ছেলের জন্যে পাত্রী খুঁজতে যাবে। বেসকেস যেকোন বিটমাস্ক ডিপির মতই হবে। মাস্কের সব বিট ১ হয়ে গেলে, অর্থাৎ সব মেয়ের বিয়ে ঠিক হয়ে গেলে ০ রিটার্ন করবে। যেহেতু কোন ছেলে কিংবা মেয়ে একাধিক বিয়ে করতে পারবে না, তাই সব মেয়ের বিয়ে ঠিক হয়ে গেছে মানেই সব ছেলের বিয়েও ঠিক হয়ে গেছে। ডিপি রিকারেন্স হবে এরকমঃ

$$dp(boy, mask) = \sum_{i=0}^{n-1} max(val[boy][girl] + dp(boy+1, newmask))$$

*newmask হল যেই মেয়ের সাথে বিয়ে ঠিক হয়েছে, সেই বিট মাস্কে সেট করার পর নতুন মাস্ক

1013 – Love Calculator

প্রব্লেম ডিস্ক্রিপশনঃ দুটো স্ট্রিং দেয়া থাকবে। দুটো জিনিস বের করতে হবেঃ

১) এমন একটা স্ট্রিং এর লেন্থ প্রিন্ট করতে হবে, যেই স্ট্রিং এ দুটো স্ট্রিং-ই সাবসিকোয়েন্স হিসেবে থাকবে। লেন্থ অবশ্যই শর্টেস্ট হতে হবে।

২) উপরের শর্ত পূরণ করে এরকম যতগুলো স্ট্রিং বানানো যায়, তার সংখ্যা।

হিন্টসঃ LCS (Longest Common Subsequence)

সল্যুশন আইডিয়াঃ ব্রেইন স্টর্মিং এর জন্যে প্রথম প্রশ্ন, সর্বোচ্চ কত লেন্থের স্ট্রিং দরকার হতে পারে, যেখানে দুটো স্ট্রিং-ই সাবসিকোয়েন্স হিসেবে থাকবে, অর্থাৎ এর বেশি লেন্থের স্ট্রিং কোনভাবেই লাগবে না।

ভেবেছো তো?! ভেবে কোন কূলকিনারা করতে না পারলে এখানে ক্লিক করো!

এখন যেই সাবসিকোয়েন্সটুকু দুটো স্ট্রিং এই কমন আছে, সেটা কিন্তু ২বার যোগ হচ্ছে, কিন্তু আমার তো লাগবে ১বার। তাহলে আমি এটা বাদ দিতে পারি। সবচেয়ে শর্ট লেন্থের স্ট্রিং পাওয়ার জন্যে সবচেয়ে বেশি লেন্থের সাবসিকোয়েন্স বাদ দেয়াই লাভজনক। তাই শর্টেস্ট স্ট্রিং এর লেন্থ হবেঃ

১ম স্ট্রিং এর লেন্থ + ২য় স্ট্রিং এর লেন্থ - LCS(string1, string2)

এবার বের করতে হবে, এই লেন্থের এক্সাক্টলি কতগুলো স্ট্রিং বানানো যায়। এটার আইডিয়াটাও অনেকটা LCS এর মতই। ডিপি স্টেট হবে (shortest_length, pos1, pos2). pos1 ১ম স্ট্রিং এর ও pos2 ২য় স্ট্রিং এর ক্যারেক্টার নির্দেশ করে। ২টা ক্যারেক্টারই যদি এক হয়, তার মানে এটা অবশ্যই শর্টেস্ট স্ট্রিং এর অন্তর্ভুক্ত হবে। আর নাহয় দুটোই যোগ করতে হবে। যখন লেন্থ পূর্ণ হবে এবং দুটো স্ট্রিং এরই সব ক্যারেক্টার বসানো হয়ে যাবে, সেক্ষেত্রে ১ রিটার্ন করব। আর নাহয় ০। এগুলোই হবে বেসকেস। কোড করার সুবিধার্থে চাইলে আরও কিছু বেসকেস যোগ করে নিতে পারো, যেমন একটা স্ট্রিং পূরণ হয়ে গেলে বাকি স্ট্রিং এর সবগুলো ক্যারেক্টারই যোগ করতে হবে, সেটা সরাসরি রিটার্ন করে দিতে পারো। নাহলে ডিপি রিকারেন্সে শর্ত দিয়ে দিতে পারো। তোমার সুবিধা অনুযায়ী যেভাবে খুশি কোড করতে পারো!

ডিপি রিকারেন্সটা হবে এরকমঃ

$$dp(length, pos1, pos2) = dp(length-1, pos1-1, pos2-1) \Rightarrow (s1[pos1] == s2[pos2])$$

$$dp(length-1, pos1-1, pos2) + dp(length-1, pos1, pos2-1) \Rightarrow (s1[pos1]! = s2[pos2])$$

1017 - Brush (III)

প্রব্লেম ডিস্ক্রিপশনঃ একটা w width এর ব্রাশ আছে তোমার কাছে, যেটা তুমি y - অক্ষের যেকোন জায়গায় বসাতে পারো এবং x - অক্ষ বরাবর মুভ করতে পারো। রুমে কিছু ধুলা আছে আর ধুলাগুলো কোথায় আছে, সেগুলোর কোঅর্ডিনেট (x,y) দেয়া আছে। ব্রাশ বসানোর পর তুমি ওই width এ x - অক্ষ বরাবর যত ধুলা আছে, একবারেই পরিষ্কার করে ফেলতে পারো। এরকমভাবে k সংখ্যকবার বা k সংখ্যক জায়গায় তুমি ব্রাশ বসাতে পারো ধুলা পরিষ্কারের জন্যে। তোমার বের করতে হবে, সর্বোচ্চ কত সংখ্যক ধুলা তুমি পরিষ্কার করতে পারো।

হিন্টসঃ সর্টিং, ন্যাপস্যাক

সল্যুশন আইডিয়াঃ প্রতিটা পার্ট পড়ার পরে অবশ্যই কিছু সময় নিয়ে সল্যুশনটা ভাবতে চেষ্টা করবে!

পার্ট - ১

পার্ট - ২

পার্ট - ৩

ডিপি রিকারেন্সটা হবে এরকমঃ

$$dp(pos, k) = max(cnt[pos] + dp(pos + step[pos], k + 1), dp(pos + 1, k))$$

এখানে cnt[pos] হচ্ছে ওই জায়গায় পরিষ্কার করলে মোট কতগুলো ধুলো পরিষ্কার হচ্ছে, আর step[pos] হচ্ছে ওই জায়গায় পরিষ্কার করার পর আমরা এরপরে কোন জায়গায় পরিষ্কার করতে যাব।

আজ এখানেই থাক! আশা করি, এই ৫টা প্রব্লেম তোমরা খুব তাড়াতাড়িই সলভ করে ফেলতে পারবে। 😀 সামনের পর্বে আরও কিছু প্রব্লেম নিয়ে হাজির হব তোমাদের সামনে!

হ্যাপি কোডিং! ☺