শাফায়েতের ব্লগ

প্রোগ্রামিং ও অ্যালগরিদম টিউটোরিয়াল

Home
অ্যালগরিদম নিয়ে যত লেখা!
আমার সম্পর্কে…

ডাইনামিক প্রোগ্রামিং এ হাতেখড়ি-১(শুরুর কথা)

এপ্রিল ২৮, ২০১২ by *শাফায়েত*

কনটেস্ট প্রোগ্রামিং করতে গিয়ে আমরা অনেক কিছুই শিখি, আমাদের গ্রাফ থিওরির অ্যালগোরিদম গুলো শিখতে হয়, জটিল সব ডাটা স্ট্রাকচার চোখের পলকে ইমপ্লিমেন্ট করতে হয়, এমনকি জ্যামিতি পর্যন্ত শিখতে হয় আমাদের। তবে প্রায় সকলেই স্বীকার করবে আমরা যেসব শিখি তারমধ্যে সবথেকে শৈল্পিক একটা বিষয় হলো ডাইনামিক প্রোগ্রামিং। এই সিরিজে আমি কিছু ক্লাসিক ডাইনামিক প্রোগ্রামিং প্রবলেম নিয়ে বিস্তারিত আলোচনা করবো যেগুলো পড়ে তুমি অনেকগুলো প্রবলেম নিজে সলভ করে ফেলতে পারবে। আমি জানি এরপর তুমি নিজেই এগিয়ে যেতে পারবে।

(এই সিরিজটি পড়া শুরুর করার আগে তোমাকে যেটা শিখতে সেটা হলো রিকার্শন। তুমি এটা শিখতে পারবে ফাহিম ভাইয়ের দারুণ এই লেখাটি পড়ে,লেখাটিতে ডাইনামিক প্রোগ্রামিং নিয়েও আলোচনা করা হয়েছে।)

ডাইনামিক প্রোগ্রামিং কোনো নির্দিষ্ট অ্যালগোরিদম নয় বরং একটি প্রবলেম সলভিং টেকনিক যা ব্যবহার করে বিভিন্ন সমস্যাকে খুবই কম সময়ের মধ্যে সলভ করা যায়। রিয়েল লাইফ অনেক প্রবলেম কম সময়ের মধ্যে একমাত্র ডাইনামিক প্রোগ্রামিং ব্যবহার করেই করা সম্ভব এবং বিশেষ করে বাংলাদেশের ন্যাশনাল কনটেস্টগুলোতে ডাইনামিক প্রোগ্রামিং প্রবলেমের একাধিক প্রবলেম সবসময় থাকে।

এখন কাজের কথায় আসি। শুরুতেই কঠিন শব্দ দিয়ে ডাইনামিক প্রোগ্রামিং এর সংজ্ঞা বললে যে কেও ভয় পেয়ে যাবে, তাই আমরা শুরু করবো ছোট্ট একটা উদাহরণ দিয়ে, তারপর কিছু ফর্মাল কথাবার্তায় যাবো। ইটালিয়ান গণিতবিদ

Leonardo Pisano Bigollo যাকে আমরা ফিবোনাচ্চি নামে চিনি খরগোশের বংশবৃদ্ধি পর্যবেক্ষণ করতে গিয়ে একটা নাম্বার সিরিজ আবিষ্কার করে বসলেন। সিরিজটি এরকম:

> ***০, ১, ১, ২, ৩, ৫, ৮, ১৩, ২১..........***

লক্ষ্য করো ১ম দুটি সংখ্যা ছাড়া প্রতিটি সংখ্যা হলো আগের দুটি সংখ্যার যোগফল। আমরা একটি ফাংশন কল্পনা করি $F(n)$ যা n তম ফিবোনাচ্চি সংখ্যা রিটার্ন করে,অর্থাত $F(n) = n$ তম ফিবোনাচ্চি সংখ্যা।
তাহলে:

> ***F(0) = 0***
>
> ***F(1) = 1***
>
> ***F(2) = F(1) + F(0) = 1***
>
> ***F(3) =F(2) + F(1) = 2***
>
> ***F(4) =F(3) + F(2) = 3***

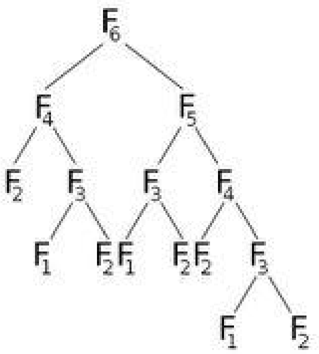
তাহলে আমরা জেনারেলভাবে বলতেই পারি:

> ***F(0) = 0***
>
> ***F(1) = 1***
>
> ***F(n) = F(n-1) + F(n-2)***

এখানে $F(0)$ এবং $F(1)$ হলো আমাদের রিকার্সিভ ফাংশনের জন্য যে বেসকেস দরকার সেটা। আমরা খুব সহজে C++ এ কোড লিখে ফিবোনাচ্চি সংখ্যা বের করতে পারি:

```cpp
int F( int n ) {
if( n == 0 ) return 0;
if( n == 1 ) return 1;
return F( n-1 ) + F( n-2 );
}
```

এখন ধরো তুমি $F(6)$ কে কল করলে। সে আবার কল করবে $F(5)$আরF(4)$ কে,এরা আরও কিছু ফাংশনকে কল করবে। আমরা ছবি আকারে দেখি কে কাকে কল করবে:



এখন খুব ভালো করে কিছু ব্যাপার লক্ষ্য করো। ধরো $F(6)$ আগে কল করছে $F(5)$ কে এবং তারপরে কল করছে $F(4)$ কে এবং সবশেষে এ দুটোর যোগফল তোমাকে দিচ্ছে। এখন ছবিতে দেখো, $F(5)$ কল দিচ্ছে $F(4)$ এবং $F(3)$ কে। যখন $F(5)$ হিসাব করা শেষ তখন অবশ্যই $F(5)$ যাদেরকে কল দিচ্ছে তাদের হিসাব করাও শেষ হয়ে গেছে,তাই নয় কি? তারমানে $F(5)$ হিসাব করতে গিয়ে $F(4)$ এবং $F(3)$ আমরা হিসাব করে ফেলেছি,এমনকি $F(2)$ ও হিসাব করে ফেলেছি । ($F(1)$ আর $F(0)$ **বেস কেস**,তাদের মান আমরা শুরু থেকেই জানি)।

এখন $F(6)$ কিন্তু $F(5)$ কে কল করার পর আবার $F(4)$ কে কল করছে। কিন্তু আমরাতো $F(4)$ এর মান হিসাব করেই ফেলেছি,কি দরকার আবার হিসাব করার? আগের মানটাই কি আমরা আবার ব্যবহার করতে পারিনা?

এখানেই আমরা একটা ছোট্ট ট্রিকস খাটাবো। কোনো একটি ফাংশনের হিসাব শেষ হয়ে গেলে আমরা একটি টেবিলে মানটি সেভ করে রাখবো। পরবর্তিতে একই ফাংশনকে আবার কল করলে আমরা পুরো হিসাব আবার না করে আগের মানটি রিটার্ন করে দিবো।

```
int dp[20];
//শুরুতে ডিপি অ্যারের সবগুলো ইনডেক্সে -১ বসিয়ে নাও
//যেমন for(int i=0;i<20;i++)dp[i]=-1; (এই কাজটা মেইন ফাংশনে করবে)
//কোনো ঘরে -১ থাকা মানে হচ্ছে ঘরটা খালি
int F( int n ) {
if( n == 0 ) return 0;
if( n == 1 ) return 1;
if( dp[n]!=-1 ) return dp[n];
else
{
dp[n] = F( n-1 ) + F( n-2 );
return dp[n];
}
}
```

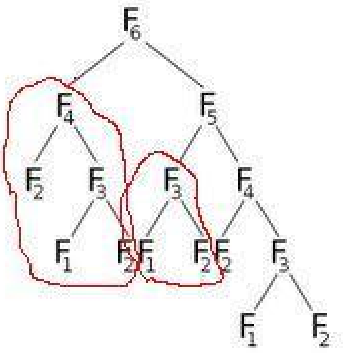
উপরের কোডে আমরা সব কাজ প্রথম কাজের মতো করছি শুধু সামান্য একটু memoization টেকনিক ব্যবহার করেছি। শুরুতে dp অ্যারের সবগুলো পজিশনে -1 রেখে নাও,তারমানে সবগুলো পজিশন খালি। তুমি যখন $F(4)$ কল করবে তখন আগে দেখো $dp[4]$ খালি নাকি,খালি হওয়া মানে এখনও হিসাব করা হয়নি,তাই $F(4)$ এর মান হিসাব করে $dp[4]$ এ রেখে রিটার্ন করে দাও। যদি খালি না হয় তারমানে আগেই $dp[4]$ এর মান হিসাব করা হয়ে গেছে!! তাহলে তুমি শুধু $dp[4]$ রিটার্ন করে দাও।

যে সহজ কাজটা করে আমরা সময় বাচিয়ে ফেললাম সেটাকেই কম্পিউটার প্রোগ্রামার কঠিন ভাষায় বলে ডাইনামিক প্রোগ্রামিং বা ডিপি। আমরা যদি মানগুলো অ্যারেতে সেভ করে না রাখতাম তাহলে একি ফাংশন বারবার কল হয়ে প্রচুর

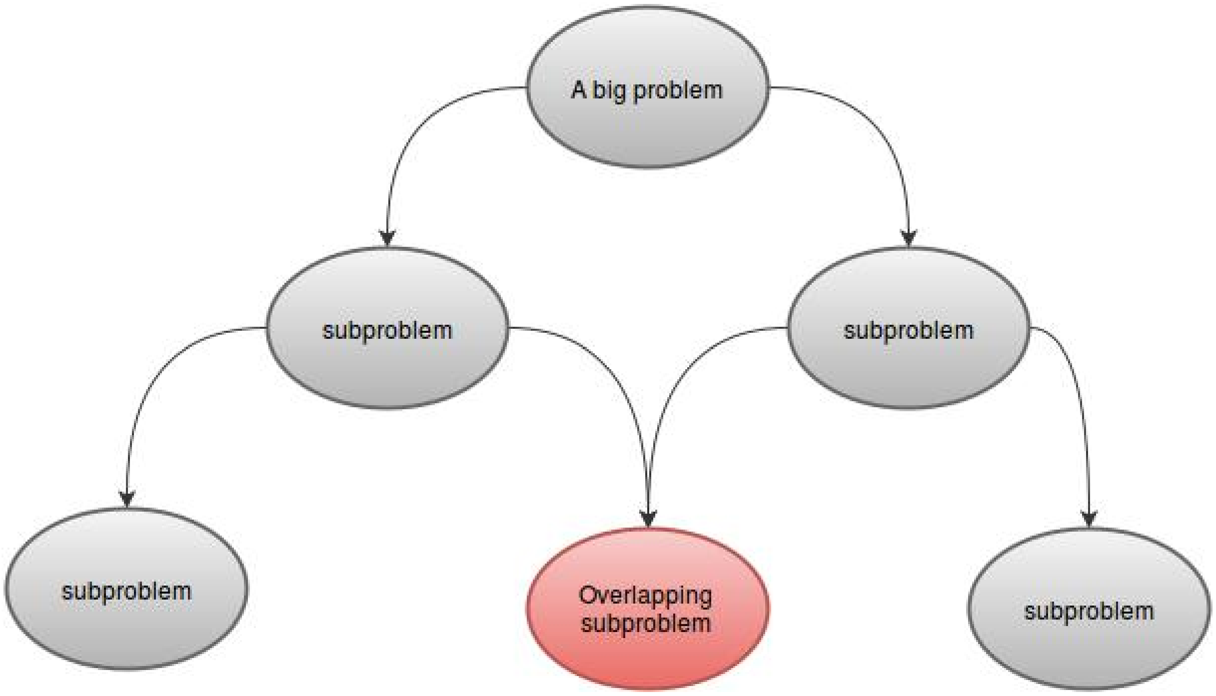
সময় নষ্ট করতো,তুমি নিজে $F(20)$ এর জন্য একবার সেভ করে আরেকবার না করে পরীক্ষা করে দেখতে পারো পার্থক্যটা কতখানি।

৩টি বৈশিষ্ট্য থাকলে একটি প্রবলেমকে ডিপি দিয়ে সলভ করা সম্ভব:
১. **subproblems:** প্রবলেমগুলোকে ছোটো এক বা একাধিক সাবপ্রবলেমে ভাগ করা যেতে হবে। যেমন $F(4)$ এর মান বের করার প্রবলেমটাকে আমরা $F(3)$ এবং $F(2)$ এই দুইভাগে ভাগ করে ফেলতে পারি। সাবপ্রবলেমগুলো মুল প্রবলেমের অনুরূপ হয়,অর্থাৎ যেভাবে মুল প্রবলেম সলভ করা যায় সেভাবেই সাবপ্রবলেম সলভ করা যায়।

২. **overlapping subproblems** থাকে। এটার মানে হলো সাবপ্রবলেমের গুলোর মধ্যে কমন অংশ থাকবে যে কারণে একই ফাংশন একাধিক ফাংশন হতে কল হবে।
যেমন $F(6)$ এবং $F(5)$ এ দুটো প্রবলেমের সাবপ্রবলেম গুলোর মধ্য $F(4)$, $F(3)$ ইত্যাদি ওভারল্যাপিং। এ কারণেই আমরা অ্যারেতে ভ্যালু সেভ করে রাখি। নিচের ছবিটি দেখো:



বামের এই লাল দিয়ে ঘেরা সাবট্রিগুলো ডানেও আছে। একাধিক স্থানে থাকার কারণেই এদের বলছি ওভারল্যাপিং সাবট্রি বা সাবপ্রবলেম। ফলে ডানপাশে আমরা একবার এগুলোকে কম্পিউট করে টেবিলে জমিয়ে রাখছি। পরে বামের অংশে এসে স্রেফ টেবিল থেকে মানটা দেখে নিচ্ছি। জেনারেলাইজ করে আরেকটি ছবি দিলাম:



ছবি: লাল নোডটা দুটো সাবপ্রবলেমকে ওভারল্যাপ করেছে

৩. **optimal substructure:** এটা ফিবোনাচ্চির উদাহরণ দিয়ে বুঝানো কঠিন। ধরো তোমাকে কোনো একটা ফাংশন $G(x)$ এর ভ্যালুকে মিনিমাইজ করতে বলেছে আর তুমি এটা জানো যে $G(x)$ নির্ভর করে $G(y)$ এবং $G(z)$ এর উপর। এখন যদি $G(y)$ এবং G(z) কে মিনিমাইজ করে $G(x)$ কে মিনিমাইজ করা যায় তাহলে প্রবলেমটির optimal substructure প্রোপার্টি আছে। এমন হতেই পারে যে $G(y)$ কে মিনিমাইজ এবং $G(z)$ কে মিনিমাইজ বা ম্যাক্সিমাইজ কিছুই না করে $G(x)$ কে মিনিমাইজ করা যায়,তাহলে optimal substructure প্রোপার্টি নেই। যদি সাবপ্রবলেমের অপটিমাল সলুশন থেকে মুল প্রবলেমের অপটিমাল সলুশন পাওয়া যায় তাহলেই শুধুমাত্র এই প্রোপার্টিটা আছে বলা যাবে।

ফিবোনাচ্চির প্রবলেমটা আসলে ডাইনামিক প্রোগ্রামিং এর খুব বেশি ভালো উদাহরন না কারণ কোনো কিছু ম্যাক্সিমাইজ বা মিনিমাইজ করতে হয়না, তবে শুরুতে বুঝানোর জন্য এটাই সবথেকে ভালো উদাহরন।

প্রথম পর্ব এখানেই শেষ। পরবর্তী পর্বে ডিপির স্টেট নিয়ে কিছু আলোচনা করা হবে এবং 2d,3d অ্যারে ব্যবহার করে ফাংশনের মান সংরক্ষন করা দেখানো হবে। এখন তোমার হোমওয়ার্ক(!!) হবে binomial coefficient nCr এর মান ডাইনামিক প্রোগ্রামিং এর সাহায্যে বের করা। রিকার্শনটি হলো nCr =(n-1)Cr + (n-1)C(r-1)। বেস কেস বের করার দায়িত্বও তোমার। পরবর্তী পর্বে সমস্যাটির সমাধান বলা হবে। hint: ২ডি টেবিলে মান সেভ করতে হবে। আর সলভ করা চেষ্টা করো এই প্রবলেমটি: http://www.lightoj.com/volume_showproblem.php?problem=1006,এটা সলভ করতে পারলে তুমি টিউটোরিয়ালটি ভালোমত বুঝেছ,না পারলে আরেকবার পড়ো এবং চিন্তা করো। ডিপি প্রবলেম সলভ করতে হলো প্রচুর চিন্তা করতে হবে,টিউটোরিয়াল পড়ে বেসিক জিনিস ছাড়া খুব বেশি কিছু শিখতে পারবেনা।
২য় পর্ব