ডাইনামিক প্রোগ্রামিং: লংগেস্ট কমন সাবসিকোয়েন্স

ডাইনামিক প্রোগ্রামিং এর সম্ভবত সবথেকে গুরুত্বপূর্ণ দুটি উদাহরণ হলো লংগেস্ট কমন সাবসিকোয়েন্স এবং এডিট ডিসটেন্স বের করা কারণ এদের অনেক প্র্যাক্টিকাল অ্যাপ্লিকেশন আছে। এই লেখাটা পড়ার আগে আমি আশা করবো তোমরা কিছু বেসিক ডাইনামিক প্রোগ্রামিং যেমন কয়েন চেঞ্জ, ন্যাপস্যাক পারো। যদি না পারো তাহলে আমার আগের লেখাগুলো দেখতে পারো। এই লেখায় লংগেস্ট কমন সাবসিকোয়েন্স বের করা, সলিউশন প্রিন্ট করাএবং সবগুলো সম্ভাব্য সলিউশন বের করা দেখবো। তুমি যদি আগেই এসব টপিক নিয়ে জানো তাহলে লেখার শেষে রিলেটেড প্রবলেম সেকশন দেখো, সেখানকার শেষ ৩টা প্রবলেম সলভ করতে কিছুটা চিন্তা-ভাবনা করা লাগবে।

> *সাবসিকোয়েন্স*

মনে করো ১টি স্ট্রিং আছে “ABC”। এই স্ট্রিংটা থেকে শূন্য, এক বা একাধিক অক্ষর মুছে দিলে যা থাকে সেটাই স্ট্রিংটার সাবসিকোয়েন্স। “ABC” এর সাবসিকোয়েন্সগুলো হলো {“ABC” , “A”,”B”,”C”,AB”,”AC”,”BC”,” “}। সবগুলো অক্ষর মুছে দিলে যা থাকে, অর্থাৎ খালি বা এম্পটি(empty) স্ট্রিং ও একটা সাবসিকোয়েন্স।

প্রতিটা অক্ষরের জন্য আমাদের হাতে ২টি অপশন ছিলো, আমরা সেটা নিতে পারি বা মুছে ফেলতে পারি। তাহলে স্ট্রিং এর দৈর্ঘ্য n হলো সাবসিকোয়েন্স থাকতে পারে 2^n টা।

> *লংগেস্ট কমন সাবসিকোয়েন্স(LCS)*

দুটি স্ট্রিং এর মধ্যে যতগুলো কমন সাবসিকোয়েন্স আছে তাদের মধ্যে সবথেকে লম্বাটাই **লংগেস্ট কমন সাবসিকোয়েন্স(LCS)**।

যেমন “HELLOM” এবং “HMLLD” এর মধ্যে “H”, “HL”, “HLL”, “HM” ইত্যাদি কমন সাবসিকোয়েন্স আছে। “HLL” হলো লংগেস্ট কমন সাবসিকোয়েন্স যা দৈর্ঘ্য ৩।

> *ব্রুটফোর্স অ্যালগোরিদম:*

আমরা ব্যাকট্র্যাকিং করে ২টা স্ট্রিং থেকে সবগুলো সাবসিকোয়েন্স জেনারেট করতে পারি। এরপরে ২টি করে সাবসিকোয়েন্স নিয়ে স্ট্রিং কম্পেয়ার করে দেখতে পারি তারা “কমন” কি না। আগেই দেখেছি মোট সাবসিকোয়েন্স হবে 2^n টা , এরপরে আবার এদের মধ্যে কম্পেয়ার করতে হবে! n এর মান ২০-২৫ পার হলেই এই অ্যালগোরিদম শেষ হতে কয়েক বছর লেগে যেতে পারে! তাই আমাদের ভালো অ্যালগোরিদম দরকার।

> *ডাইনামিক প্রোগ্রামিং:*

ডাইনামিক প্রোগ্রামিং এর মূল কথা কি মনে আছে? প্রবলেমটাকে ছোট ছোট ভাগ করো, সেই ভাগ গুলো আগে সলভ করো আর সেগুলো জোড়া লাগিয়ে বড় প্রবলেমটার সমাধান বের করে ফেলো। কয়েন চেঞ্জের সময় আমরা প্রতিটা কয়েন নিয়ে বা ফেলে দিয়ে বাকি কয়েনগুলোর জন্য সলভ করেছি। এখানে প্রতিটা ক্যারেক্টারের জন্য সেই কাজ করবো!

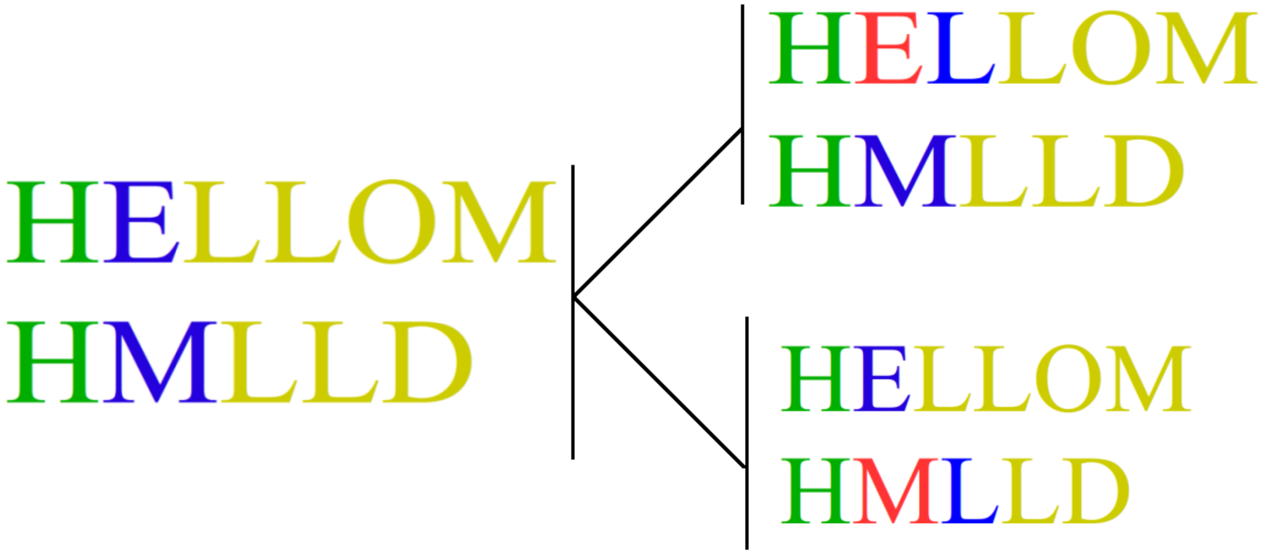
মনে করো “HELLOM” এবং “HMLLD” এদের মধ্যে LCS বের করতে চাও। শুরুতে তুমি আছো ২টা স্ট্রিং এরই প্রথম ক্যারেক্টারে।

নীল রঙ দিয়ে বুঝাচ্ছে তুমি কোন স্ট্রিং এর কোন ক্যারেকটারে আছো। **নীল রঙের ক্যারেকটার থেকে শুরু করে বাকি স্ট্রিংটুকুর জন্য তোমাকে প্রবলেমট সলভ করতে হবে।**

এখন লক্ষ্য করো নীল রঙের অক্ষর দুটি একই। তারমানে তুমি কিছু চিন্তা না করেই এই অক্ষরটাকে LCS এর মধ্যে ঢুকিয়ে দিতে পারো এবং বাকি স্ট্রিংটুকুর জন্য রিকার্সিভলি প্রবলেমটা সলভ করতে পারো।

আমরা H অক্ষরটাকে নিয়ে নিয়েছি। এখন ২টি স্ট্রিং এই আমরা ২য় অক্ষরে আছি। নীল অক্ষর থেকে বাকি স্ট্রিংটুকুর জন্য আমরা এখন সলভ করবো। তাহলে প্রবলেমটা ছোট হয়ে গেলো, আমাদের “ELLOM”, আর “MLLD” এর মধ্যের LCS বের করতে হবে।

এবার নীল রঙের অক্ষরদুটি একই না। তারমানে **অন্তত ১টাকে বাদ দিয়ে** LCS হিসাব করতে হবে। আমরা একবার উপর থেকে E বাদ দিয়ে হিসাব করবো আরেকবার নিচ থেকে M বাদ দিয়ে হিসাব করবো।



তারমানে LCS(“ELLOM”,”MLLD”) বের করতে আমরা LCS(“LLOM”,”MLLD”) বের করবো এবং LCS(“ELLOM”,”LLD”) বের করবো। এই দুইটার মাঝে যেটা বড় সেটা নিবো!

সবগুলো স্টেট ছবি একে দেখাবো না, তুমি নিশ্চয়ই বুঝে গেছো আমাদের কাজ কি হবে। মনে করো ফাংশন calcLCS(i,j) প্রথম স্ট্রিং এর i তম ক্যারেকটার এবং ২য় স্ট্রিং এর j তম ক্যারেকটার থেকে বাকি স্ট্রিং এর LCS বের করে। অর্থাৎ i,j হলো ১ম আর ২য় স্ট্রিং এর নীল রঙের ক্যারেকটার। আর স্ট্রিং দুটি হলো A আর B। তাহলে ফাংশনটা ডিফাইন করা যায় এভাবে:

> *calcLCS(i,j)=1+calcLCS(i+1,j+1) যদি A[i]==B[j] হয়*
>
> *calcLCS(i,j)=max(calcLCS(i+1,j) , calcLCS(i,j+1) যদি A[i]!=B[j] হয়*

রিকার্সনটা রানটাইম ইরোর দেয়ার আগে শেষ হবে না যদি না বেস-কেস দাও। কোন স্টেটের জন্য আমরা হিসাব না করেই উত্তর বলে দিতে পারবো? প্রতি ধাপে স্ট্রিংগুলো ছোট হতে হতে কোন স্ট্রিং যদি “এম্পটি” হয়ে যায় তাহলে তার সাথে অন্য যেকোন স্ট্রিং এর LCS ০ হবে।

> *calcLCS(i,j)=০ যদি A[i]==NULL বা B[j]==NULL হয়।*

একটু খেয়াল করলেই দেখবে একই ফাংশন বারবার কল হবে। তাই আমাদের মেমরাইজেশন করতে হবে যেটা আমরা অন্যসব ডিপি প্রবলেমে করি। পুরে কোডটা হতে পারে এরকম:

LCS
C++

```
#define MAXC 1000
char A[MAXC],B[MAXC];
int lenA,lenB;
int dp[MAXC][MAXC];
bool visited[MAXC][MAXC];
int calcLCS(int i,int j)
{
    if(A[i]=='\0' or B[j]=='\0') return 0;
    if(visited[i][j])return dp[i][j];

    int ans=0;
    if(A[i]==B[j]) ans=1+calcLCS(i+1,j+1);
    else
    {
        int val1=calcLCS(i+1,j);
        int val2=calcLCS(i,j+1);
        ans=max(val1,val2);
    }
    visited[i][j]=1;
    dp[i][j]=ans;
    return dp[i][j];
}
int main() {
    scanf("%s%s",A,B);
    lenA=strlen(A);
    lenB=strlen(B);
    printf("%d\n",calcLCS(0,0));
    return 0;

}
```

> *সলিউশন প্রিন্ট:*

এখন কথা হলো LCS কত বড় সেটাতো পেলাম, কিন্তু স্ট্রিংটা পাবো কিভাবে? এটাও খুব সহজ, calcLCS ফাংশনটা যে পথে এগিয়েছে সে পথে এগিয়ে গেলেই আমরা স্ট্রিংটা পেয়ে যাবো। আমরা আরেকটা ফাংশন লিখতে পারি, মনে করো ফাংশনটা হলো printLCS(i,j)। এখানেও i,j দিয়ে স্ট্রিং দুটির ইনডেক্স বুঝাচ্ছে। এখন A[i]==B[j] হলে আমরা অক্ষরটিকে প্রিন্ট করে (i+1,j+1) স্টেটে চলে যাবো। আর A[i]!=B[j] হলে আগেই হিসাব করা ডিপি অ্যারে থেকে dp[i+1][j] আর dp[i][j+1] এর মান দেখে সামনে আগাবো, যেদিকে গেলে ম্যাক্সিমাম দৈর্ঘ্য পাওয়া যাবে সেদিকে যাবো। নিচের কোডটা দেখলে পরিষ্কার হবে:

Printing LCS
C++

```
string ans;
void printLCS(int i,int j)
{
    if(A[i]=='\0' or B[j]=='\0'){
        cout<<ans<<endl;
        return;
    }
    if(A[i]==B[j]){
        ans+=A[i];
        printLCS(i+1,j+1);
    }
    else
    {
        if(dp[i+1][j]>dp[i][j+1]) printLCS(i+1,j);
        else printLCS(i,j+1);
    }
}
```

অর্থাৎ আমাদের calcLCS ফাংশন যেদিকে গিয়ে সর্বোচ্চ মান পেয়েছে আমরা সে পথ ধরেই আগাচ্ছি।

এখন প্রশ্ন আসতে পারে যে সলিউশনতো একাধিক থাকতে পারে, সবগুলো পাবো কিভাবে? যেমন “hello” আর “loxhe” এই দুইটা স্ট্রিং এর LCS দুইটা, “he” এবং “lo”। সবগুলো সলিউশন চাইলে ব্যাকট্র্যাকিং করতে হবে। ধরো printAll(i,j) ফাংশনটা সবগুলো সলিউশন প্রিন্ট করে। আগের মতো যদি A[i]==B[j] তাহলে (i+1,j+1) এ যাবো। আর যদি A[i]!=B[j] হয় তাহলে dp[i+1][j] আর dp[j][i+1] এর যেটা বড় সেদিকে আগাবো, পার্থক্য হলো **যদি dp[i+1][j]==dp[i][j+1] হয় তাহলে আমরা দুইদিকেই আগাবো।**

ALL LCS
C++

```
string ans;
void printAll(int i,int j)
{
    if(A[i]=='\0' or B[j]=='\0'){
        cout<<ans<<endl;
        return;
    }
    if(A[i]==B[j]){
        ans+=A[i];
        printAll(i+1,j+1);
        ans.erase(ans.end()-1); //Delete last character
    }
    else
    {
        if(dp[i+1][j]>dp[i][j+1]) printAll(i+1,j);
        else if(dp[i+1][j]<dp[i][j+1]) printAll(i,j+1);
        else
        {
            printAll(i+1,j);
            printAll(i,j+1);
        }
    }
}
```

কোডটা অনেকটা আগেরমতোই। “ans.erase(ans.end()-1);” এরকম একটা অতিরিক্ত লাইন দেখতে পাচ্ছো। এটার কাজ স্ট্রিং এর শেষ ক্যারেকটারটা মুছে ফেলা। এটা কেন করছি? এই লাইনটা মুছে ফেললে সমস্যা কি হবে? এটা তুমি চিন্তা করে বের করো, লাইনটা মুছে চালিয়ে দেখো কি

হয়।

> *কমপ্লেক্সিটি:*

স্ট্রিং দুটির দৈর্ঘ্য n এবং m হলে calcLCS() ফাংশনটি মোট n*m টা স্টেটে থাকতে পারে। তাহলে **কমপ্লেক্সিটি O(n*m)**।

> *প্র্যাকটিস প্রবলেম:*

Longest Common Subsequence
The Twin Towers
History Grading
Is Bigger Smarter?

> *রিলেটেড প্রবলেম ১: এডিট ডিসটেন্স(লেভেল-১)*

তোমাকে দু্টি স্ট্রিং A,B দেয়া আছে। তুমি **শুধুমাত্র A স্ট্রিংটার** উপর ৩টা অপারেশন করতে পারো, কোনো একটা ক্যারেকটার বদলে দিতে পারো, কো ক্যারেকটার মুছে ফেলতে পারো, যেকোন পজিশনে নতুন ক্যারেকটার ঢুকাতে পারো। তারমানে চেঞ্জ, ডিলিট, ইনসার্ট হলো তোমার ৩টা অপারেশন। এখন তোমার কাজ মিনিমাম অপারেশনে A স্ট্রিংটাকে B বানানো। যেমন “blog” কে “bogs” বানাতে তুমি l মুছে ফেলে স্ট্রিং এর শেষে s ইনসার্ট করতে পারো।
(হিন্টস: LCS এর মতোই দুইটা ইনডেক্স i,j কে স্টেট রাখতে হবে। এখন তুমি চিন্তা করো স্ট্রিং A থেকে কোন ক্যারেকটার মুছে ফেললে i,j এর পরিবর্তন কিরকম হবে। ঠিক সেভাবে বাকি ২টি অপারেশনের জন্য কিভাবে i,j পরিবর্তন হবে সেটা বের করো)
প্রবলেম লিংক

> *রিলেটেড প্রবলেম ২: লেক্সিকোগ্রাফিকালি মিনিমাম লংগেস্ট কমন সাবসিকোয়েন্স(লেভেল-২)*

২টি স্ট্রিং এর যতগুলো LCS আছে তাদের মধ্য থেকে **লেক্সিকোগ্রাফিকালি মিনিমাম** টা বের করতে হবে। “hello” আর “loxhe” এর মধ্যে **লেক্সিকোগ্রাফিকালি** মিনিমাম LCS হলো “he”।
প্রবলেম লিংক

> *রিলেটেড প্রবলেম ৩: লংগেস্ট কমন ইনক্রিসিং সাবসিকোয়েন্স(লেভেল-৩)*

এই প্রবলেমে শুধু এমন সব সাবসিকোয়েন্স বিবেচনা করতে হবে যারা ছোট থেকে বড় সাজানো। এদের মধ্যে লংগেস্ট কমন সাবসিকোয়েন্সটা বের করতে হবে। প্রবলেম লিংক

> *রিলেটেড প্রবলেম ৪: ভিন্ন ভিন্ন লংগেস্ট কমন সাবসিকোয়েন্স(লেভেল-৩)*

২টি স্ট্রিং এর মধ্যে কয়টা ভিন্ন ভিন্ন LCS আছে বলতে হবে। স্ট্রিং এর দৈর্ঘ্য সর্বোচ্চ ১০০০ হতে পারে, ব্যাকট্র্যাকিং করে করা যাবে না। প্রবলেম লিংক

হ্যাপি কোডিং!