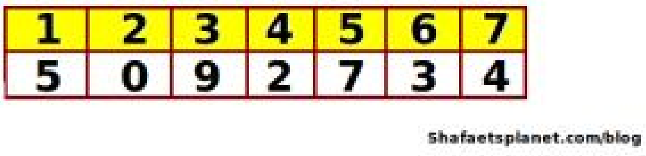
Home
অ্যালগরিদম নিয়ে যত লেখা!
আমার সম্পর্কে...

ডাইনামিক প্রোগ্রামিং এ হাতেখড়ি-৪

আগস্ট ২৫, ২০১২ by *শাফায়েত*

০-১ ন্যাপস্যাক,কয়েন চেঞ্জ প্রবলেমে আশা করি তোমার এখন দক্ষতা এসে গিয়েছে। এই পর্বে আমরা দেখবো LIS, একই সাথে দেখবো কিভাবে ডিপিতে সলিউশন প্রিন্ট করতে হয় ,এটা নিয়ে তোমাদের অনেকেরই সমস্যা হয়েছে বলে জানিয়েছো। এছাড়া আগের পর্বে light oj 1231 প্রবলেমটি সলভ করতে বলেছিলাম,আমার সলিউশন পাবে লেখার একদম শেষে।

প্রথমেই শুরু করি LIS এবং এটার সলিউশন প্রিন্ট করা দিয়ে। LIS হলো Longest increasing subsequence। মনে করো তোমাকে একটি অ্যারে বা sequence দেয়া আছে:



1	2	3	4	5	6	7
5	0	9	2	7	3	4

Shafaetsplanet.com/blog

এই অ্যারে থেকে কিছু সংখ্যা মুছে দিয়ে এবং অর্ডার ঠিক রেখে আমরা বিভিন্ন subsequence পেতে পারি। যেমন:

> *5 7*
>
> *5 9 2*
>
> *0 3 4*

top

0 2 3 4

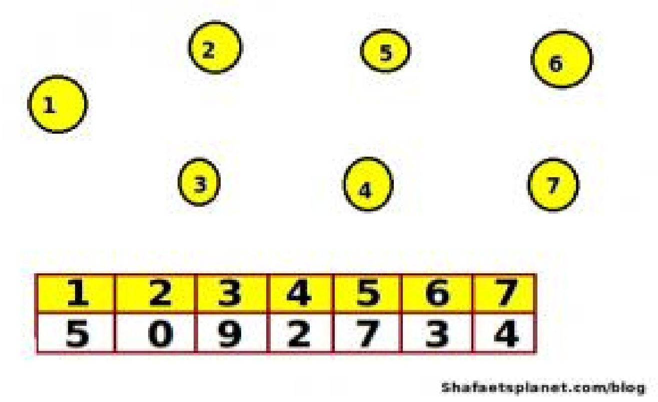
7 3

5 0 9 2 7 3 4 (শুন্যটি সংখ্যা বাদ দেয়া হয়েছে)

….

একটা sequence এ n টি সংখ্যা থাকলে subsequence থাকতে পারে মোট 2^n টি(কেন??)। increasing subsequence**(IS)** এ প্রতিটা সংখ্যা তার আগের সংখ্যাটির থেকে বড় হবে। উপরে ১ম , ৩য়, ৪র্থ subsequence গুলো increasing। যতগুলো IS আছে তারমধ্যে সবথেকে বড়টা খুজে বের করাই আমাদের লক্ষ্য। উপরে ৪নম্বরটির length ৪ এবং সবগুলো IS এর মধ্যে এটাই সব থেকে বড়।

প্রবলেমটাকে গ্রাফ দিয়ে মডেলিং করলে খুব সহজে সলভ করা যায়। গ্রাফ থিওরি না জানলেও কোনো সমস্যা হবেনা,গ্রাফ দেখাচ্ছি খালি বোঝার সুবিধার জন্য। প্রথমেই মনে করি অ্যারের প্রতিটি ইনডেক্স হলো একটি করে নোড:



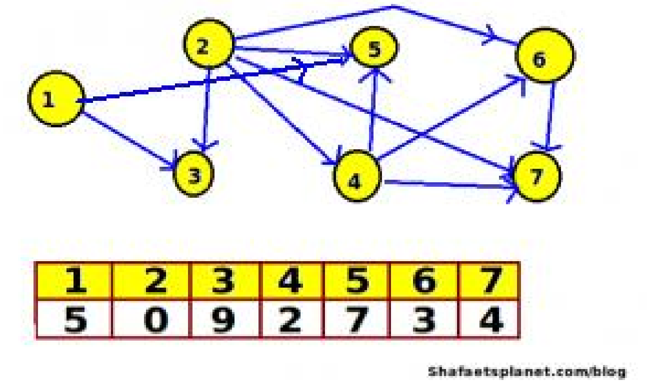
1	2	3	4	5	6	7
5	0	9	2	7	3	4

Shafaetsplanet.com/blog

এখন দুটি শর্ত খেয়াল করো। u নম্বর নোড থেকে v নম্বরে নোডে যাওয়া যাবে যদি:

> *১. v>u হয়। কারণ আমাদের মূল sequence এর অর্ডার ঠিক রাখতে হবে।*
>
> *২. value[v]>value[u] হয়, কারণ sequence টা increasing হতে হবে।*

তাহলে ১ থেকে ২ এ যেতে পারবোনা কারন value[1] < value[2],তবে ১ থেকে ৩ এ যেতে পারবো। তাহলে আমরা গ্রাফের edge গুলো আকি:



1	2	3	4	5	6	7
5	0	9	2	7	3	4

Shafaetsplanet.com/blog

দেখতে হিজিবিজি হলেও ভয় পাবার কিছু নাই,উপরের শর্ত দুটি মেনে গ্রাফটি আকা হয়েছে। গ্রাফের যেকোনো পাথ অনুসরণ করলে আমরা একটা ইনক্রিসিং সিকোয়েন্স পাবো। যেমন গ্রাফে ২-৬-৭ পাথে গেলে ০-৩-৪ সাবসিকোয়েন্সটা পাবো। তাহলে বুঝতেই পারছো সবথেকে লম্বা পথটাই আমার সলিউশন, যতদূরে যেতে পারবো সিকোয়েন্স তত বড় হবে, এক্ষেত্রে একটি path এ যে কয়টি নোড আছে সেটাই পাথের দৈর্ঘ্য।

মনে করি আমাদের একটা ফাংশন আছে longest(u) যেটা u নম্বর নোড থেকে longest path রিটার্ন করে। যেমন longest(6)=2 কারণ ৬ নম্বর নোড থেকে খালি ৬-৭ এই পাথে যাওয়া যায়। এখন লক্ষ করো:

> “ *কোনো নোড থেকে longest path হবে* ***সেই নোড থেকে যেসব নোডে যাওয়া যায় সেগুলো থেকে longest path এর maximum + ১।***

আমরা সহজেই একটা রিকার্সিভ রিলেশন বের করে ফেলতে পারি:

> “ *longest(2)=1+max(longest(3),longest(4),longest(5),longest(6),longest(7))*
>
> *longest(1)=1+max(longest(3),longest(5))*
>
> *longest(3) =1 (এটা একটা বেসকেস কারণ ৩ থেকে কোথাও যাওয়া যায়না)*
>
> …………
>
> *অর্থাৎ u থেকে v1,v2,v3…,vk নোডে যাওয়া গেলে:*
>
> *longest(u)=1+max(longest(v1),longest(v2)….,longest(vk))*

এখন আমরা সহজেই কোড করে প্রতিটা নোড থেকে longest path বের করে ফেলতে পারি:

```
#define mx 1000
int n=7;
int value[]={-100000,5,0,9,2,7,3,4};
int dp[mx],dir[mx];
int longest(int u)
{
    if(dp[u]!=-1) return dp[u];
    int maxi=0;
    for(int v=u+1;v<=n;v++) //১ম শর্ত,v>u
    {
        if(value[v]>value[u]) //২য় শর্ত, value[v]>value[u]
        {
            if(longest(v)>maxi) //সর্বোচ্চ মানটা নিবো
            {
                maxi=longest(v);
                                dir[u]=v;

            }
        }
    }
    dp[u]=1+maxi; //১ যোগ হবে কারণ u নম্বর নোডটাও পাথের মধ্যে আছে
    return dp[u];
```

top

```
}
int main()
{
    READ("in");
    memset(dp,-1,sizeof dp);
        memset(dir,-1,sizeof dir);
    int LIS=0,start;
    for(int i=1;i<=n;i++)
    {
        printf("longest path from: %d\n",longest(i));
        if(longest(i)>LIS)
        {
            LIS=longest(i);
            start=i;
        }
    }
    printf("LIS = %d Starting point %d\n",LIS,start);

    return 0;
}
```

longest(u) প্রতিটা নোড u থেকে longest path রিটার্ন করে,এদের মধ্যে আবার যেটা ম্যাক্সিমাম সেটাই আমাদের LIS। এই সলিউশনটা কমপ্লেক্সিটি O(n^2)(আরো ভালো একটা সলিউশন আছে বাইনারী সার্চ ব্যবহার করে)। এখন সলিউশন প্রিন্ট করার পালা।

আসলে মুল কাজটা আমরা উপরের কোডেই করে ফেলসি। dir[] নামের একটি অ্যারে রেখেছি যে প্রতিটা নোডের জন্য যে ডিরেকশনে সর্বোচ্চ মান পাওয়া যায় সেটা সেভ করে রাখে। কোনো নোড থেকে অন্য কোথাও যাওয়া না গেলে ডিরেকশন হবে -১। এখন সলিউশন ফাংশন হবে এরকম:

```
void solution(int start)
{
    while(dir[start]!=-1)
    {
        printf("index %d value %d\n",start,value[start]);
        start=dir[start];
    }
}
```

অর্থাৎ ডিরেকশন অ্যারে দেখে আমরা সামনে যাবো যতক্ষণ যাওয়া যায়।

এখানে লক্ষ্য করার কিছু বিষয় হলো উপরের গ্রাফটিতে কোনো সাইকেল নেই দেখে আমরা ডিপি করতে পেরেছি,এধরণের গ্রাফকে বলা হয় directed acyclic graph বা ড্যাগ। কোনো প্রবলেমকে তুমি যদি ড্যাগ এ কনভার্ট করতে পারো তাহলে খুব ভালো সম্ভাবনা আছে যে প্রবলেমটি ডিপি চালিয়ে সলভ করা যাবে। গ্রাফে সাইকেল থাকলে longest path একটি np-complete প্রবলেম,অর্থাৎ পলিনমিয়াল সলিউশন জানা নেই। এখন একটি প্রশ্ন: একটা sequence এর কয়টি LIS আছে সেটা বের করতে বলা হলে কি করবে? চিন্তা করে জানাও :-)।

আমরা জানি ০-১ন্যাপস্যাকে স্টেট থাকে ২টা,সেক্ষেত্রে ডিরেকশন অ্যারের স্টেটও হবে ২টা। নিচের কোডটা দেখো:

```
int dir[][]={{-1}};
int dp[][]={{-1}};
int func(int i,int w) //i নম্বর আইটেম নিয়ে চেষ্টা করা হচ্ছে,w ওজনের জিনিস নেয়া হয়েছে
{
    .........
    //BASE CASE
```

top

```
    .........

    if(w+weight[i]<=CAP) //i নম্বর জিনিসটি নিবো
        profit1=cost[i]+func(i+1,w+weight[i])
    else
        profit1=0;
    profit2=func(i+1,w) // i নম্বর জিনিসটি নিবো না
    if(profit1>profit2){dir[i][w]=1; return dp[i][w]=profit1;}
    else {dir[i][w]=2; return dp[i][w]=profit2;}
}
```

ন্যাপস্যাকের কোডটাকে সামন্য পরিবর্তন করে ডিরেকশন রাখা হয়েছে। আমাদের starting state হলো (i=1,w=0),কারণ শুরুতে আমরা ১ নম্বর জিনিসটা নিয়ে ট্রাই করবো এবং এখন পর্যন্ত নেয়া জিনিসের ওজন ০। ডিপি শেষ হবার পর যদি:

> *dir[i][w] = ১ হয় তাহলে i নম্বর জিনিসটি নিবো, i প্রিন্ট করে আমরা চলে যাবো (i+1,w+weight[i]) স্টেটে।*
>
> *dir[i][w] = ২ হয় তাহলে i নম্বর জিনিসটি নিবো না,i প্রিন্ট না করেই আমরা চলে যাবো (i+1,w) স্টেটে।*
>
> *dir[i][w] = -১ হলে আমরা থেমে যাবো।*

এটাই হলো ডিপির সলিউশন প্রিন্টের মূল কথা। ডিপি অ্যারের পাশাপাশি ডিরেকশন অ্যারেতে সেভ করে রাখবো কোন স্টেট থেকে কোন স্টেটে যাচ্ছি সেটা। এরপর starting state থেকে ডিরেকশন অনুযায়ী আগাতে থাকবো আর প্রিন্ট করবো।

এক স্টেট থেকে একাধিক স্টেটে গেলে, আমরা সেভ করবো যে স্টেটে গেলে
ম্যাক্সিমাইজ বা মিনিমাইজ বা যে স্টেট থেকে ভ্যালিড আউটপুট পাওয়া যায় শুধু সেটা

shafaetsplanet.com/blog

```
struct next
{
    int a,b;
    next(int _a,int _b){a=_a,b=_b;}
    next(){}
}direction[100][100];
int dp[100][100];
int call(int a,int b)
{
    ............
    ............
    int ret1=call(a+1,b);
    int ret2=call(a,b+1);
    if(ret1>ret2) {
        dp[a][b]=ret1;
        direction[a][b]=next(a+1,b);
    }
    else {
        dp[a][b]=ret2;
        direction[a][b]=next(a,b+1);
    }
    return dp[a][b];
}
```

ডিপি অ্যারেতে রিটার্ণ ভ্যালু সেভ করছি,
আর ডিরেকশন অ্যারেতে কোন দিকে যাচ্ছি সেটা সেভ করছি!

প্রিন্ট করার সময় আমরা ডিরেকশন অ্যারের দেখানো
পথে শুধু সামনে এগিয়ে যাবো!

```
void go(int a,int b)
{
    //Dont forget to return from function when base case reached
    //otherwise you'll fall into infinite loop
    int next_a=dir[a][b].a;
    int next_b=dir[a][b].b;
    if(next_a!=a) puts("Took a");
    else puts("Took b");
    go(next_a,next_b);
}
```

top

আজকের পর্ব এখানেই শেষ। light oj 1231 এর সলিউশন: http://pastebin.com/vmpWp0g1, mod করার অংশটা বুঝতে না পারলে এই লেখাটা দেখো। সলভ করার জন্য প্রবলেম:

- Diving for gold
- Sense of Beauty
- Bicolored Horses
- Testing the CATCHER
- How Many Dependencies?
- Longest Path

পরের পর্ব-বিটমাস্ক ডিপি
সবগুলো পর্ব

ফেসবুকে মন্তব্য

0 comments

0 Comments Sort by Newest

Add a comment...

Facebook Comments plugin

Powered by Facebook Comments

Posted in অ্যালগোরিদম/প্রবলেম সলভিং ? Tagged LIS, ডাইনামিক প্রোগ্রামিং, ডিপি

30,175 বার পড়া হয়েছে

গ্রাফ থিওরি: স্টেবল ম্যারেজ প্রবলেম
মিনিমাম ভারটেক্স কভার প্রবলেম
top