**ডাইনামিক প্রোগ্রামিং**

**লেখক: শাফায়েত**

[**ডাইনামিক প্রোগ্রামিং এ হাতেখড়ি-১(শুরুর কথা)**](http://www.shafaetsplanet.com/planetcoding/?p=1022)

**এই সিরিজটি পড়া শুরুর করার আগে তোমাকে যেটা শিখতে সেটা হলো রিকার্শন।**

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**

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:

int F( int n ) {

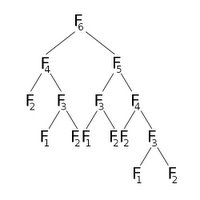
if( n == 0 ) return 0;

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) আর F(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 ডাইনামিক প্রোগ্রামিং ব্যবহার করবো। কোনো একটি ফাংশনের হিসাব শেষ হয়ে গেলে আমরা একটি ফাংশনে মানটি সেভ করে রাখবো। পরবর্তিতে একই ফাংশনকে আবার কল করলে আমরা পুরো হিসাব আবার না করে আগের মানটি রিটার্ন করে দিবো।

int dp[20];

//initialize dp with -1 using a For loop or While loop or Memset or any other way!

//For Example: for(int i=0;i<20;i++)dp[i]=-1; (do it in main function)

//if a array cell contains -1 that means that cell is empty

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 -১ 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 কাজটা করে আমরা সময় বাচিয়ে ফেললাম সেটাই ডাইনামিক প্রোগ্রামিং। আমরা যদি মানগুলো dp 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।

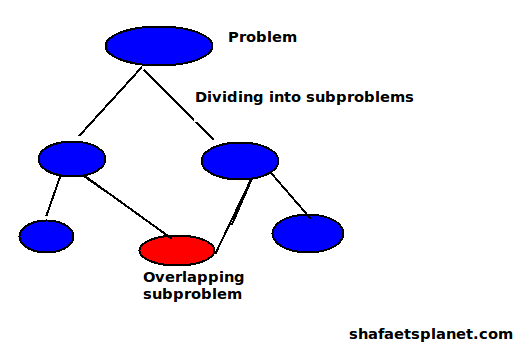
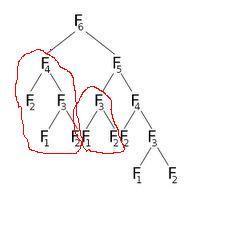
ডাইনামিক প্রোগ্রামিং কে সংক্ষেপে বলা হয় ডিপি।

**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 similiar হয়,অর্থাত 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। ছবিটা সুন্দর না হলেও যা বুঝাতে চেয়েছি সেটা বুঝা যাচ্ছে। 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।

[**ডাইনামিক প্রোগ্রামিং এ হাতেখড়ি-২**](http://www.shafaetsplanet.com/planetcoding/?p=1072)

[১ম পর্বে](http://www.shafaetsplanet.com/planetcoding/?p=1022) আমরা জেনেছি ডাইনামিক প্রোগ্রামিং কাকে বলে,প্রবলেমে কি রকমের বৈশিষ্ট্য থাকলে সেটা ডাইনামিক প্রোগ্রামিং এর সাহায্যে সমাধান করা যায়। আমরা দেখেছি ডিপি দিয়ে কিভাবে ফিবোনাচ্চি সংখ্যার রিকার্শনের রানটাইম অনেক কমিয়ে আনা যায়। তবে ডিপি এমন একটা জিনিস যে এতকিছু জেনেও তুমি কিছুই সমাধান করতে পারবেনা যদিনা খুব ভালো করে প্র্যাকটিস করো আর চিন্তা করো। তবে এটা শুনে ভয়ের একদমই কিছু নেই, প্র্যাকটিস করতে থাকলে কিছুদিন পর দেখবে অনেক সহজেই রিকার্সিভ ফাংশন বের করে ডিপি প্রবলেম সলভ করে ফেলতে পারছো,আমার কাজ হলো তোমাকে শুরু করিয়ে দেয়া। সব শেষে [hexabonacci](http://www.lightoj.com/volume_showproblem.php?problem=1006) (loj-1006) নামের একটি প্রবলেম সলভ করতে দিয়েছিলাম,আশা করি সবাই প্রবলেমটি খুব সহজেই সমাধান করে ফেলেছে। সমাধান করতে না পারলে আমি বলবো আগের পর্বটা আরেকবার খুব ভালোভাবে পড়ে ফেলতে আর কোনো জায়গায় না বুঝলে মন্তব্য অংশে জানাতে। প্রবলেমটির সাথে ফিবোনাচ্চি প্রবলেমটার খুব একটা পার্থক্য নেই বলে আমি সমাধান নিয়ে আলোচনা করছিনা।

আরেকটি কাজ দিয়েছিলাম,সেটা হলো nCr এর মান বের করা। এটার সমাধান নিয়ে এখন আলোচনা করবো। কলেজে থাকতে হয়তো পড়েছো:

  ^nC_{r} =^{(n-1)}C_{r} + ^{(n-1)}C_{(r-1)}  

আমরা এই রিলেশনটি ব্যবহার করে nCr এর মান বের করবো, সুত্রের প্রমান এখানে অপ্রাসঙ্গিক হয়ে যাবে। আমরা যদি মনে করি nCr(n,r) হলো একটি ফাংশন যা n আর r এর মানকে প্যারামিটার হিসাবে গ্রহণ করে তাহলে আমরা উপরের রিলেশন তাকে এভাবে লিখতে পারি:

nCr(n,r) =nCr(n-1,r) + nCr(n-1,r-1)

এবার হয়তো জিনিসটা আরো স্পষ্ট হয়েছে তোমার কাছে। কিন্তু রিকার্শনটা শেষ হবে কখন? অর্থাৎ বেস কেস কি? আমরা জানি nCr(n,1)=n এবং nCr(n,n)=1 । এ দুটি শর্ত বেস কেস হিসাবে ব্যবহার করলে কোডটা হবে:

#define i64 long long

i64 nCr(int n,int r)

{

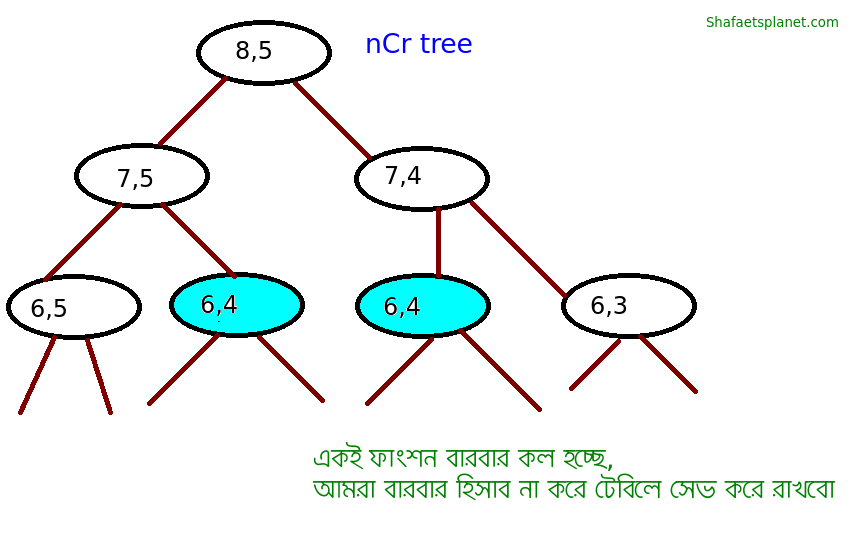
if(r==1) return n;

if(n==r) return 1;

return nCr(n-1,r)+nCr(n-1,r-1);

}

ফাংশন কল গুলো লক্ষ্য করলে দেখতে পাবে ফিবোনাচ্চির মতো এখানেও একটি ফাংশন বার বার কল হচ্ছে:



আমরা এই অতিরিক্ত ফাংশন কল সহজেই এড়াতে পারি আগের মতো একটা টেবিল রেখে। পার্থক্য হলো এবার টেবিলটা হবে ২-ডি,আর কোনো পার্থক্য নেই। আগে টেবিলটার সব ঘরে -১ রেখে নিবো,-১ দিয়ে বুঝাচ্ছে ঘরটি খালি আছে। আগের লেখায় অনেকে initialisation অংশটা বুঝতে পারেনি তাই এবার পুরো কোডটা দিচ্ছি।

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 | i64 dp[70][70];  i64 nCr(int n,int r)  {  if(r==1) return n;  if(n==r) return 1;  if(dp[n][r]!=-1) return dp[n][r]; //ভ্যালু টেবিলে থাকলে নতুন করে হিসাব করা দরকার নেই,ভ্যালুটা রিটার্ণ করে দাও  else{  dp[n][r]=nCr(n-1,r)+nCr(n-1,r-1); //ভ্যালু টেবিলে সেভ করে রাখো  return dp[n][r];  }      }  int main()  {  //init dp table with -1  for(int i=0; i< 70;i++)  for(int j=0;j < 70;j++)  dp[i][j]=-1;  printf("%d\n",nCr(20,2));    } |

আমাদের সবগুলো ডিপি কোডই মোটামুটি একটা ফরমেট মেনে চলবে,সেটা এরকম:

int/double/long long call(int x1,int x2........,int xn)

{

CHECK BASE CASES,RETURN IF REACHED ANY OF THE BASE CASES

CHECK TABLE,RETURN IF VALUE IS ALREADY COMPUTED

ELSE

{

COMPUTE VALUE RECURSIVELY AND SAVE IN A N-Dimensional TABLE/ARRAY

RETURN VALUE

}

}

nCr এর ক্ষেত্রে প্যারামিটার ছিলো n এবং r। বুঝতেই পারছো যতগুলো প্যারামিটার থাকবে মান টেবিলে সেভ করার জন্য তত ডাইমেনশনের টেবিল লাগবে। স্টেট বেশি লাগলে মেমরিও বেশি লাগে। অ্যারের প্রতিটি ডাইমেনশনের সাইজ প্রয়োজনমতো প্যারামিটার অনুসারে কমবেশি হবে।

প্যারামিটারকে আমরা সাধারণত বলে থাকি স্টেট(State)। সম্ভবত ডিপিতে সবথেকে গুরুত্বপূর্ণ অংশ হলো স্টেট বুঝতে পারা। ধরো তোমাকে ৫০মিনিটের মধ্যে একটা ক্লাস ধরতে হবে। এখন রাস্তায় তুমি কোন অবস্থায় আছো সেটা আমি জানতে পারবো যদি তুমি আমাকে দুটি তথ্য দাও: তুমি এই মুহুর্তে কোন জায়গায় আছো আর তুমি বাসা থেকে বের হবার পর কয় মিনিট পার হয়েছে। যেমন হয়তো তুমি ফার্মগেটে আছো আর বাসা থেকে বের হবার পর ২০ মিনিট পার হয়েছে। তুমি কি রঙের জামা পড়েছো বা তুমি কোন জুতা পড়েছো এটা কিন্তু এখানে গুরুত্বপূর্ণ না তাই এটা “স্টেট” এর মধ্যে পড়েনা।

কোনো এক রাতে তুমি চুরি করতে বের হলে!! বন্ধুর বাসায় জানালা দিয়ে ঢুকে দেখলে প্রচুর জিনিসপত্র,কিন্তু তোমার চুরি করার থলেতে জায়গা আছে মাত্র ১০ইউনিট,এর বেশি নিলে থলে ছিড়ে যাবে। প্রতিটা জিনিসের ভর আছে আর একেক জিনিসের মুল্যও একেকরকম।

১. মানিব্যাগ: ১ইউনিট,১২০টাকা  
২. কোরম্যানের-বই ভর: ৭ইউনিট,৪০০টাকা  
৩. ডিভিডি-কালেকশন ভর: ৪ইউনিট,২৮০ টাকা  
৪. ফেলুদা-সমগ্র: ৩ইউনিট,১৫০টাকা  
৫. ফুটবল: ভর: ৪ইউনিট,২০০টাকা

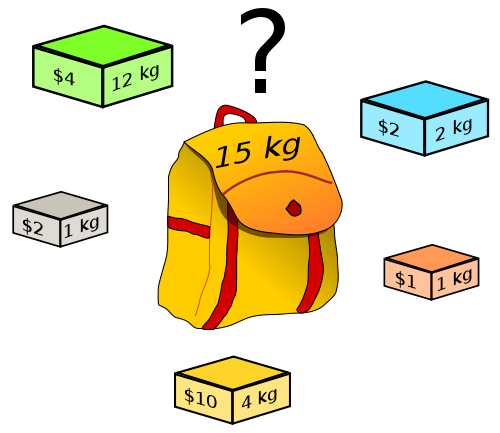
কোনো জিনিস নিলে পুরোটাই নিতে হবে,৪টি ডিভিডির ২টি তুমি নিতে পারবেনা,ফেলুদা সমগ্রের অর্ধেক ছিড়ে আনতে পারবেনা।

প্রথমদিন চুরি করতে গিয়েছো এই জন্য কিছু না ভেবেই তুমি ফটাফট দামি জিনিসগুলো ভরতে থাকলে। প্রথমেই তুমি ৪০০টাকার কোরম্যানের বই নিয়ে নিলে,তারপর ১৫০টাকার ফেলুদা সমগ্র নিয়ে বাসায় ফিরে আসলে,তোমার লাভ হলো ৫৫০ টাকা। বাসায় এসে হিসাব করে দেখলে তুমি যদি [লোভীর মতো (greedy)](http://www.c3.lanl.gov/mega-math/gloss/compute/greedy.html) দামী জিনিসগুলো আগে না নিয়ে একটু ভেবে-চিন্তে নিতে তাহলে ২৮০টাকার ডিভিডি,২০০টাকার ফুটবল আর ১২০টাকার মানিব্যাগ নিয়ে ফিরতে পারতে,তোমার লাভ হতো ৬০০টাকা।

[greedy](http://www.c3.lanl.gov/mega-math/gloss/compute/greedy.html) অ্যলগোরিদমে অপটিমাল রেজাল্ট না পাওয়ায় তুমি চিন্তা করলে সবরকমের কম্বিনেশনে চেষ্টা করবে, প্রতিবার একটা করে জিনিস থলেতে ভরবে আর দেখবে আর কত লাভ করা যায়,প্রয়োজনে জিনিসটা থলে থেকে নামিয়ে রেখে আরেকটি নিয়ে চেষ্টা করবে। এবং সুবিধার জন্য তুমি লিস্টের সিরিয়াল অনুযায়ী জিনিস নিয়ে চেষ্টা করবে, ৩ নম্বর জিনিসের পরে ১ নম্বর জিনিস নিতে চেষ্টা করবেনা।

কোনো সময় তোমার অবস্থা বুঝাতে ২টা তথ্যই যথেষ্ট।

১. তুমি এখন কত নম্বর জিনিস ট্রাই করছো।  
২. এখন পর্যন্ত তুমি কত ইউনিট জিনিস নিয়েছো।



তুমি একটা ফাংশন লিখে ফেললে যেটা জিনিসপত্রের লিস্ট থেকে অপটিমাল রেজাল্ট বের করে দিতে পারে। ফাংশনটির রিটার্ন টাইপ আর প্যারামিটার হবে এরকম:

i=current\_index  
w=weight taken  
int func(int i,int w)

মনে করি i নম্বর বস্তুটির ভর হলো weight[i] আর মূল্য cost[i]। আর ব্যাগের capacity=CAP।

প্রতিবার তুমি i নম্বর জিনিসটি নিতে পারো যদি ব্যাগে জায়গা থাকে, অথবা তুমি i নম্বর জিনিসটি না নিয়ে i+1 তম জিনিস ট্রাই করতে পারো।

i নম্বর জিনিসটি যদি তুমি নাও তাহলে লাভ হবে **cost[i] + পরবর্তি স্টেটে লাভ**, তাহলে আমরা cost[i] যোগ করে পরবর্তি স্টেটে চলে গিয়ে কত লাভ হয় সেটা হিসাব করবো।

//w+weight[i] হলো i তম জিনিস নেবার পর মোট ওজন,এটা কখনোই CAP এর বেশি হতে পারবেনা।

if(w+weight[i]<=CAP)

profit1=cost[i]+func(i+1,w+weight[i])

else

profit1=0; //নেয়া গেলোনা,নিলে ব্যাগ ছিড়ে যাবে

//cost[i] হলো i তম জিনিসটার দাম

//func(i+1,w+weight[i]) = আমরা i+1 তম জিনিস নিয়ে ট্রাই করবো

i নম্বর জিনিসটি যদি তুমি না নাও তাহলেও লাভ বেশি হতে হবে তাই সেটাও আমাদের হিসাব করতে হবে:

profit2=func(i+1,w)

//নতুন ওজন যোগ হচ্ছেনা,i+1 তম বস্তু নিয়ে ট্রাই করছি।

লক্ষ্য করো এক্ষেত্রে কারেন্ট প্রফিট বা ওজনের কোনো পরিবর্তন হচ্ছেনা, কোনো কিছু না নিয়েই পরের স্টেটে গিয়ে দেখছি লাভ কত হবে।

তাহলে সহজেই বোঝা যাচ্ছে অপটিমাল রেজাল্টের জন্য আমাদের profit1 আর profit2 এর মধ্যে বড়টা নিতে হবে,সেই মানটা আমরা রিটার্ণ করে দিবো।

return max(profit1,profit2)

তাহলে তুমি যদি main থেকে func(1,0) কল করো তাহলে রিকার্শন তোমাকে রেজাল্ট বের করে দিবে। func(1,0) কল করার কারণ হলো শুরুতে তুমি ১ নম্বর জিনিসটা নিয়ে ট্রাই করবে এবং শুরুতে ব্যাগে সম্পূর্ণ খালি। যখন সবগুলো জিনিস নিয়ে ট্রাই করা হয়ে যাবে তখন রিকার্শন শেষ হবে,অর্থাৎ

n টা বস্তু থাকলে বেসকেস হবে if(i==n+1) return 0

শুন্য রিটার্ন করছি কারণ সবকিছু নেয়া হয়ে গেলে আর প্রফিট করা সম্ভবনা।

তোমার বোঝার সুবিধার জন্য সম্পূর্ণ একটা কোড দিয়ে দিলাম:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 | #define MAX\_N 100  #define MAX\_W 1000  int n;  int dp[MAX\_N+1][MAX\_W+1];  int weight[MAX\_N+1];  int cost[MAX\_N+1];  int CAP;  int func(int i,int w)  {  if(i==n+1) return 0; //সব কিছু নেয়া হয়ে গেছে  if(dp[i][w]!=-1) return dp[i][w]; //এই স্টেটটা আগেই হিসাব করে এসেছি  int profit1=0,profit2=0;  if(w+weight[i]<=CAP)  profit1=cost[i]+func(i+1,w+weight[i]);//যদি i তম জিনিসটা নেয়া যায় তাহলে লাভের পরিমাণ profit1      profit2=func(i+1,w); //যদি জিনিসটা না নেই তাহলে লাভ profit2  dp[i][w]=max(profit1,profit2); //বেশি লাভ যেটায় হবে সেটাই আমরা নিবো  return dp[i][w];  }  int main()  {    freopen("in","r",stdin);  memset(dp,-1,sizeof(dp));  scanf("%d%d",&n,&CAP);  for(int i=1;i<=n;i++)  {  scanf("%d%d\n",&weight[i],&cost[i]);  }  printf("%d\n",func(1,0));      } |

এই কোডে উপরের উদাহরণের ইনপুটটা নিচের মতো করে দিলে দিলে ৬০০ আউটপুট আসবে

5 10 //৫ টা জিনিস, ১০ ক্যাপাসিটি  
1 120 //প্রতিটা জিনিসের ওজন এবং দাম  
7 400  
4 280  
3 150  
4 200

একটু চিন্তা করলেই দেখবে আগের প্রবলেমগুলোর মতো এটাতেও একই ফাংশন বারবার কল হয়,তাই ২-ডি অ্যারেতে মানগুলো সেভ করতে হবে। অ্যারের সাইজ হবে [n][cap] যেখানে n=কয়টা জিনিস আছে, এই উদাহরণে n=5। তবে মেমরির ব্যবহার অনেক কমিয়ে আনা যায় কাজটা রিকার্শনে না করে iterative করলে,সেটা আমরা পরে দেখবো।

এটাই হলো ক্লাসিকাল [0-1 knapsack](http://en.wikipedia.org/wiki/Knapsack_problem) প্রবলেম,ন্যাপস্যাক শব্দটির অর্থ হলো থলে। ০-১ নাম দেয়ার কারণ হলো তুমি কোনো জিনিস অর্ধেক নিতে পারবেনা,হয় পুরোটা নিবে অথবা নিবেনা। fractional-knapsack বলে আরেক ধরণের প্রবলেম আছে,সেটার আলোচনা অন্য কোনোদিন হবে। ন্যাপস্যাকে সলিউশন প্রিন্ট করা শিখতে [সিরিজের চতুর্থ পর্বটা](http://www.shafaetsplanet.com/planetcoding/?tag=%E0%A6%A1%E0%A6%BF%E0%A6%AA%E0%A6%BF) দেখো।

এখন তোমার এখন কাজ হলো এটার সম্পূর্ণ কোড ইমপ্লিমেন্ট করা। [uva 10130](http://uva.onlinejudge.org/external/101/10130.html) প্রবলেমটি সলভ করতে চেষ্টা করো।

ডাইনামিক প্রোগ্রামিং এ ভালো করার একমাত্র উপায় হলো প্রচুর চিন্তা করা। তাই আমি বেশিভাগ সলিউশন ১০০% না দিয়ে চিন্তা করার জন্য কিছু অংশ রেখে দিবো। তোমাকে অনুরোধ করবো পরবর্তি পর্ব পড়ার আগে অবশ্যই আগের পর্বের সমস্যাগুলো নিয়ে খুব ভালো করে অন্তত ১দিন চিন্তা করবে।

**মন্তব্য**

মুকিত

কোডটি করেছি… Accepted ও হয়েছে…কিন্তু রান টাইম তো অনেক বেশি !!! 1.088 sec !!!কমানো যায় না ??? contest time এ Time Limit পাবো নাতো ???

শাফায়েত

হয়তো কমানো যায়। ডিপি অ্যারে বার বার initialise না করে একটা বুলিয়ান অ্যারে দিয়ে কাজটা করলে রানটাইম কমবে। যেমন:  
if(dp[n][r]!=-1) return dp[n][r]; এটার যায়গায় লেখা যায়:  
if(vis[n][r]==1) return dp[n][r];  
vis[n][r]=1;  
এখানে vis একটা বুলিয়ান অ্যারে,তাই memset করতে অনেক কম সময় লাগবে।

Rahat

0-1 knapsack এ ,finally যেসব জিনিস নেয়া হয়েছে, ওই গুলো print দেবার জন্য কিভাবে store করব?????

শাফায়েত

এই লেখাটায় সেটা নিয়ে লিখেছি: <http://www.shafaetsplanet.com/planetcoding/?p=1211>

rajib

#include<stdio.h>

#include<math.h>

int main()

{

long int num,k=1,i,paste;

while(scanf("%ld",&num)==1)

{

if(num<0) break;

for(i=0;; i++)

{

if(pow(2,i)>=num)

{

paste=i;

break;

}

}

printf("Case %ld: %ld\n",k,paste);

k++;

}

return 0;

}

IT

এই দুইটা লাইন বুজি নাইঃ  
memset(dp,-1,sizeof(dp));

শাফায়েত  
memset দিয়ে ডিপি অ্যারের প্রতিটি পজিশনে -1 বসিয়েছি।

IT

‘memset’ ব্যাবহার করলে কোডব্লক্সে এই লাইটা আসেঃ  
D:\Books\Code\Online Judge\New-1 knapsack.c|31|warning: incompatible implicit declaration of built-in function ‘memset’|  
আর আমার আঊটপুট 600 না দেখিয়ে 0 দেখাচ্ছে।  
কোড লিঙ্কঃ <http://ideone.com/oPDnE3>

মুহিম মুক্তাদীর

লাইট ওযে এর ১০০৬ নাম্বার প্রবলেমটা accepted হচ্ছেনা… ওভারফ্লো হচ্ছে এই ইনপুট এর জন্যঃ ১ ৯৯৯৯ ৯৯৯৯ ৯৯৯৯ ৯৯৯৯ ৯৯৯৯ ৯৯৯৯৯ ৯৯৯৯

আমার কোডঃ  
<http://pastebin.com/Yn2CGiCd>

মুহিম মুক্তাদীর

UVA 10130 supersale problem acceptd হয়েছে… runtime 0.792s :)

Rajkin Hossain

aro fast accepted hobea jodi apni memozition pordhoti na korea bottom-up solution vabea solve koren… :)

Rajkin Hossain

ohh no i am sorry for knapsack based problem bottom-up is not faster the memozition…

[**ডাইনামিক প্রোগ্রামিং এ হাতেখড়ি-৩**](http://www.shafaetsplanet.com/planetcoding/?p=1158)

[আগের পর্বগুলো](http://www.shafaetsplanet.com/planetcoding/?tag=%E0%A6%A1%E0%A6%BF%E0%A6%AA%E0%A6%BF) পড়ে থাকলে তুমি এখন ডাইনামিক প্রোগ্রামিং নিয়ে বেসিক ব্যাপারগুলো কিছুটা শিখে গিয়েছো,যত প্রবলেম সলভ করবে তত দক্ষতা বাড়বে। ডিপিতে আসলে কোনো নির্দিষ্ট অ্যালগোরিদম না থাকায় আমাদের চিন্তা করতে হয় অনেক বেশি,একেকটি ডিপি প্রবলেম একেক ধরণের, তবে তুমি যদি ন্যাপস্যাক,কয়েন চেঞ্জের মতো ক্লাসিক কিছু ডিপি প্রবলেমের সলিউশন জানো তাহলে তুমি বুঝতে পারবে কিভাবে তোমার চিন্তাকে এগিয়ে নিয়ে যেতে হবে,কিভাবে ডিপির স্টেট নির্ধারণ করতে হবে,তখন তুমি নতুন ধরণের ডিপি প্রবলেমও সলভ করে ফেলতে পারবে। আমি এরই মধ্যে nCr নির্ণয় আর ০-১ ন্যাপস্যাকের ডিপি সলিউশন নিয়ে আলোচনা করেছি,আরো কিছু ক্লাসিক বা স্ট্যান্ডার্ড প্রবলেম নিয়ে সামনে আলোচনা করবো।

আগের পর্বে [uva-10130 supersale](http://uva.onlinejudge.org/external/101/10130.html) প্রবলেমটি সলভ করতে বলেছিলাম। প্রবলেমটি সহজ ০-১ ন্যাপস্যাক প্রবলেম,আশা করি সবাই করতে পেরেছো। আমার সলিউশনটি এরকম:

#define mem(x,y) memset(x,y,sizeof(x));

int dp[1002][102];

int Weight[1002],Cost[1002];

int cap,n;

int call(int i,int w)

{

if(i==n+1) return 0;

if(dp[i][w]!=-1) return dp[i][w];

int profit1;

if(w+Weight[i]<=cap)profit1=Cost[i]+call(i+1,w+Weight[i]);

else profit1=0;

int profit2=call(i+1,w);

int ret=max(profit1,profit2);

return dp[i][w]=ret;

}

int main()

{

int t;

cin>>t;

while(t--)

{

cin>>n;

for(int i=1;i<=n;i++)cin>>Cost[i]>>Weight[i];

int q,ans=0;

cin>>q;

while(q--) {

mem(dp,-1);

cin>>cap;

int ret=call(1,0);

ans+=ret;

}

cout<<ans<<endl;

}

return 0;

}

আগের পোস্টে যেভাবে বলেছি ঠিক সেভাবেই কোড করা হয়ছে,বুঝতে সমস্যা হবার কথা না। প্রবলেমে বলা হয়েছে “Print out the maximal value of goods which we can buy with that family.”, তাই q জন ফ্যামিলি মেম্বারের জন্য ন্যাপস্যাক চালিয়ে ম্যাক্সিমাম বের করে ভ্যালুগুলো ans এর সাথে যোগ করে দিয়েছি। যদি একটি আইটেম একবারের বেশি নেয়া যেতো তাহলে কোডে কোন অংশটা পরিবর্তন হতো চিন্তা করে বলতে পারবে? এক্ষেত্রে ১২ নম্বর লাইনে i+1 এর জায়গায় i বানিয়ে দিলেই হবে, তাহলে কোনো একটি আইটেম নেবার পর আবারো সেই আইটেম নিতে চেষ্টা করবে। কোনো লাইন বুঝতে সমস্যা হলে কমেন্ট করে জানাও বা আমাকে একটা [মেইল](http://www.shafaetsplanet.com/planetcoding/?page_id=2) করো।

এখন আমরা দেখবো কয়েন চেঞ্জ প্রবলেম। আসলে ন্যাপসাক শেখার পরে কয়েন চেঞ্জ তুমি এমনিই পারবে তারপরেও লিখছি যাতে ব্যাপারগুলো আর পরিস্কার হয়।



তোমার কাছে কিছু কয়েন আছে যাদের মূল্য 5,8,11,15,18 ডলার। প্রতিটা কয়েন অসীম সংখ্যকবার আছে,তুমি যেকোনো কয়েন যতবার ইচ্ছা নিতে পারো। তাহলে তোমার coin অ্যারেটা হতে পারে এরকম:

int coin[]={5,8,11,15,18};

এখন তোমাকে এই কয়েনগুলো নিয়ে নির্দিষ্ট কোনো ভ্যালু বানাতে হবে। ধরি সংখ্যাটি হলো “make”। make=18 হলে আমরা ৫+৫+৮ এভাবে ১৮ বানাতে পারি। তোমাকে বলতে হবে কয়েনগুলো দিয়ে ভ্যালুটি বানানো যায় নাকি যায়না। greedy অ্যালগোরিদম এখানে কাজ করবেনা (কেনো করবেনা?)।  
প্রথমেই আমরা চিন্তা করি ডিপিতে স্টেট কি হবে। আমরা একটি একটি কয়েন নিয়ে সংখ্যাটি বানাতে চেষ্টা করতে থাকবো। তাহলে এই মুহুর্তে কোন কয়েন নিচ্ছি সেটা স্টেট রাখতে হবে,আর আগে যেসব কয়েন নিয়েছি সেগুলোর মোট ভ্যালু কত সেটা রাখতে হবে। ফাংশনটির নাম call হলে প্রোটোটাইপ হবে:

int call(int i,int amount)

এরপর অনেকটা আগের মতোই কাজ। প্রথমে i নম্বর কয়েন নিতে চেষ্টা করবো:

if(amount+coin[i]<=make) ret1=call(i,amount+coin[i]);  
else ret1=0;

এখানে i+1 কল না করে আবার i কল করছি কারন এক কয়েন অনেকবার নেয়া সম্ভব। যদি এক কয়েন একাধিক বার নেয়া না যেতো তাহলে i+1 কে কল দিতাম। amount+coin[i] যদি make এর থেকে বড় হয় তাহলে কয়েনটি নেয়া সম্ভবনা। কয়েন যদি না নেই তাহলে আমরা পরবর্তী কয়েনে চলে যাবো:

ret2=call(i+1,amount);

ret1 আর ret2 এর কোনো একটি true হলেও make বানানো যাবে। তাহলে সবশেষে লিখবো:

return ret1|ret2;

আর বেসকেস হবে হলো, যদি সব কয়েন নিয়ে চেষ্টা করার পর make বানানো যায় তাহলে return 1,অন্যথায় return 0। সম্পূর্ণ কোড:

int coin[]={5,8,11,15,18}; //value of coins available

int make; //our target value

int dp[6][100];

int call(int i,int amount)

{

if(i>=5) { //All coins have been taken

if(amount==make)return 1;

else return 0;

}

if(dp[i][amount]!=-1) return dp[i][amount]; //no need to calculate same state twice

int ret1=0,ret2=0;

if(amount+coin[i]<=make) ret1=call(i,amount+coin[i]); //try to take coin i

ret2=call(i+1,amount); //dont take coin i

return dp[i][amount]=ret1|ret2; //storing and returning.

}

int main()

{

// freopen("in","r",stdin);

while(cin>>make)

{

memset(dp,-1,sizeof(dp));

cout<<call(0,0)<<endl;

}

return 0;

}

এখন যদি তোমাকে বলা হতো যে কোনো একটি ভ্যালু কতবার বানাতে হবে বলতে হবে তাহলে কি করতে? যেমন ১৮ বানানো যায় ২ ভাবে, এক্ষেত্রে ret1|ret2 রিটার্ন না করে ret1+ret2 রিটার্ন করে দাও,তাহলে যত ভাবে বানানো যায় সবগুলো যোগ হয়ে যাচ্ছে। [uva-674](http://uva.onlinejudge.org/external/6/674.html) প্রবলেমটিতে এটাই করতে বলা হয়েছে,ঝটপট সলভ করে ফেলো।

উপরের মতো করে কোড করে tle খাওয়ার পর এখন একটি অপটিমাইজেশন দেখো। আমরা প্রতিবার make ইনপুট নেয়ার পর ডিপি অ্যারে নতুন করে initialize বা ক্লিয়ার করেছি। যদি সেটা করতে না হতো তাহলে অনেক কম সময় লাগতো, কারণ একই মান বারবার হিসাব করা লাগবেনা। কিন্তু ক্লিয়ার করতে হচ্ছে কারণ ফাংশনটি বাইরের একটি গ্লোবাল ভ্যারিয়েবলের উপর নির্ভরশীল, “if(amount==make)return 1;” এই লাইনটাই ঝামেলা করছে, make এর মান প্রতি কেসের জন্য আলাদা, তাই প্রতিবার নতুন করে সব হিসাব করতে হচ্ছে। আমরা যদি make কে একটা স্টেট হিসাবে রাখি তাহলে কাজ হয় কিন্তু স্টেট বিশাল হয়ে যায়। এর থেকে আমরা সমস্যাটাকে উল্টায় ফেলি। মনে করো তোমার কাছে শুরুতে ২০ টাকা আছে, বিভিন্ন অ্যামাউন্টের কয়েন দান করে দিয়ে তোমাকে শুন্য টাকা বানাতে হবে। কোডটা এবার হবে এরকম:

int coin[]={5,8,11,15,18}; //value of coins available

int make=18; //we will try to make 18

int dp[6][100];

int call(int i,int amount)

{

if(i>=5) { //All coins have been taken

if(amount==0)return 1;

else return 0;

}

if(dp[i][amount]!=-1) return dp[i][amount]; //no need to calculate same state twice

int ret1=0,ret2=0;

if(amount-coin[i]>=0) ret1=call(i,amount-coin[i]); //try to take coin i

ret2=call(i+1,amount); //dont take coin i

return dp[i][amount]=ret1|ret2; //storing and returning.

}

int main()

{

// freopen("in","r",stdin);

memset(dp,-1,sizeof(dp));

while(cin>>make)

{

cout<<call(0,make)<<endl;

}

return 0;

}

খেয়াল করে দেখো ঠিক আগের মতোই কাজ করেছি, শুধু যোগ করার জায়গায় বিয়োগ করে make থেকে শুন্য বানানোর চেষ্টা করেছি। লাভটা হলো এখন ফাংশনটি কোনো পরিবর্তনশীল গ্লোবাল ভ্যারিয়েবলের উপর নির্ভর করেনা, তাই মেইন ফাংশনে ডিপি অ্যারে লুপের মধ্যে ক্লিয়ার করা দরকার নাই। প্রতিবার কয়েন একই থাকছে বলে এই ট্রিকসটা কাজ করছে, কয়েনের মান পরিবর্তন হলে কাজ করবেনা। ডিপির প্রবলেমে অনেকসময় টেস্টকেস অনেক বেশি দেয় যাতে বার বার ক্লিয়ার করলে টাইম লিমিট পাস না করে।

এখন আমরা complexity আর মেমরি হিসাব করি। ডিপি অ্যারের সাইজ হবে [কয়েন সংখ্যা][সর্বোচ্চ যত ভ্যালু বানাতে হবে]। যেহেতু ভিতরে কোনো লুপ চলছেনা ভিতরে তাই complexity ও একই হবে,O(num\_of\_coin+make)।

লাইটওজেতে [1231 – Coin Change (Ioj)](http://www.lightoj.com/volume_showproblem.php?problem=1231) প্রবলেমে কিছু কয়েন দিয়ে একটি ভ্যালু কয়ভাবে বানানো যায় সেটা বের করতে বলেছে,তবে প্রতিটি কয়েন সর্বোচ্চ কয়বার ব্যবহার করা যাবে সেটা বলা আছে,i নম্বর কয়েন Ci বার ব্যবহার করা যাবে। এই কন্ডিশনটা দুইভাবে তুমি হ্যান্ডেল করতে পারো। ৩য় একটি স্টেট রাখতে পারো taken\_i যেটা বলে দিবে i নম্বর তুমি কয়বার নিয়েছো,Ci বার ব্যবহার হয়ে গেলে পরবর্তী কয়েনে চলে যাও।

int call(int i,int taken\_i,int amount)

২য় উপায় হলো ফাংশনের ভিতরে Ci পর্যন্ত একটি লুপ চালিয়ে কয়েনটি যতবার নেয়া সম্ভব ততবার নিয়ে অ্যামাউন্টটি বানাতে চেষ্টা করো,এক্ষেত্রে মেমরি কম লাগবে। তাহলে আজকের ২য় কাজ হলো এই প্রবলেমটা সলভ করা,না পারলে পরবর্তী পর্বে কোড পাবে তবে চেষ্টা না করে অবশ্যই কোড দেখবেনা। যদি modulo নিয়ে সমস্যা হয় তাহলে মডুলার অ্যারিথমেটিক নিয়ে [আমার লেখাটা](http://www.shafaetsplanet.com/planetcoding/?p=936) দেখতে পারো।

কয়েন চেঞ্জ প্রবলেমের আরেকটি নাম হলো subset sum problem,কারণ কিছু নম্বরের সেট থেকে একটি সাবসেট আমাদের নিতে হয় যেটার যোগফল এক নির্দিষ্ট ভ্যালুর সমান।

আরেকধরণের প্রবলেম নিয়ে অল্প আলোচনা করে পর্বটি শেষ করবো। তোমাকে একটি ২ডি অ্যারে দেয়া হলো:

-1 2 5  
4 -2 3  
1 2 10

তুমি শুরুতে আছো (০,০) সেলে। তুমি শুধু ৩দিকে যেতে পারো:

(i+1,j)  
(i+1,j-1)  
(i+1,j+1)

প্রতিটি সেলে গেলে তোমার পয়েন্টের সাথে ওই সেলের সংখ্যাটি যোগ হয়। তুমি সর্বোচ্চ কত পয়েন্ট বানাতে পারবে? উপরের অ্যারেতে সর্বোচ্চ পয়েন্ট ৭(-১+-২+১০)। ডিপিটা একদম সহজ কিন্তু খুবই গুরুত্বপূর্ন,অনেক প্রবলেম সলভ করা যায় এভাবে। তুমি স্টেট রাখো এখন কোন সেলে আছো সেটা। প্রতিটা সেল থেকে তুমি ৩ দিকে যেতে চেষ্টা করো। যেদিকে গেলে সর্বোচ্চ পয়েন্ট পাবে সেটা রিটার্ণ করো। বেসকেস হলো যদি অ্যারের বাইরে চলে যাও তাহলে return 0। কোডটা এরকম:

#define inf 1<<28

int mat[][10]={

{-1, 2, 5},

{4, -2, 3},

{1 , 2 ,10,}

};

int dp[10][10];

int r=3,c=3;

int call(int i,int j)

{

if(i>=0 && i<r and j>=0 and j<c) //if still inside the array

{

if(dp[i][j]!=-1) return dp[i][j];

int ret=-inf;

//try to move to 3 direction,also add current cell's point

ret=max(ret,call(i+1,j)+mat[i][j]);

ret=max(ret,call(i+1,j-1)+mat[i][j]);

ret=max(ret,call(i+1,j+1)+mat[i][j]);

return dp[i][j]=ret;

}

else return 0; //if outside the array

}

int main()

{

// READ("in");

mem(dp,-1);

printf("%d\n",call(0,0));

return 0;

}

৩দিকে মুভ করার কন্ডিশন কেনো দেয়া হয়েছে? adjacent ৪টি সেলে মুভ করতে দিলে সমস্যা কোথায় হতো? তাহলে একটি সাইকেল তৈরি হতো,একটি ফাংশনের কাজ শেষ হবার আগেই রিকার্শনে ঘুরে ফাংশনটি আবার কল হতো,এই সলিউশন কাজ করতো না। এখানে আমরা যে ৩দিকে মুভ করছি তাতে কোনো সাইকেল তৈরি হচ্ছেনা। গ্রাফ নিয়ে কিছুটা জানলে তুমি প্রতিটা সেলকে গ্রাফের একটি নোড কল্পনা করতে পারো,আর সেটা থেকে ৩টা নোডে যেতে পারছি সেগুলোকে edge দাও। তাহলে একটি directed গ্রাফ তৈরি হবে যেটায় কোনো সাইকেল নেই। একে বলে directed acyclic graph বা DAG। সাইকেল না থাকায় DAG এর উপর সহজেই ডিপি চালানো যায়। আমরা সামনের পর্বগুলোতে আরো কিছু উদাহরণ দেখবো এটার।

[1004 – Monkey Banana Problem(loj)](http://www.lightoj.com/volume_showproblem.php?problem=1004) এই প্রবলেমটা অনেকটা উপরের প্রবলেমের মতো,সলভ করতে কোনো সমস্যা হবেনা। তুমি যদি bfs/dfs পারো তাহলে [uva-11331](http://uva.onlinejudge.org/external/113/11331.html) প্রবলেমটি সলভ করে ফেলো,(হিন্ট:বাইকালারিং+ন্যাপস্যাক)। এছাড়া এগুলো ট্রাই করো:

[Uva 11137: Ingenuous Cubrency](http://uva.onlinejudge.org/external/111/11137.html)(Coin change)  
[Codeforces 118D: Caesar’s Legions](http://www.codeforces.com/problemset/problem/118/D)(4 state)  
[Light oj 1047: Neighbor House](http://www.lightoj.com/volume_showproblem.php?problem=1047)   
[Timus 1017: Staircases](http://acm.timus.ru/problem.aspx?space=1&num=1017)

চেষ্টা করো সবগুলো প্রবলেম সলভ করতে,আটকে গেলে সমস্যা নাই,চিন্তা করতে থাকো,এছাড়া ন্যাপস্যাক আর কয়েন চেঞ্জ সম্পর্কিত যতগুলো পারো প্রবলেম সলভ করে ফেলো,যেহেতু তুমি এখন বেসিক পরবর্তী পর্বগুলোতে ডিটেইলস কমিয়ে নতুন নতুন প্রবলেম আর টেকনিক নিয়ে বেশি আলোচনা করবো। শুভকামনা থাকলো :)।

**মন্তব্য**

নওরীন

while(q–)

{  
mem(dp,-1);  
cin>>cap;  
int ret=call(1,0);  
ans+=ret;  
}

এই খানে এসে confused হয়ে গেছি। q আর ans এর কাজ কি?

শাফায়েত

প্রবলেমে বলা আছে “Next line contains one integer (1<=G<=100) it’s the number of people in our group.”। q টা এখানে G। তারপর বলেছে “. Print out the maximal value of goods which we can buy with that family.” । q  
জন ফ্যামিলি মেম্বারের জন্য ম্যাক্সিমামটা বের করে পুরোটা যোগ করে দিয়েছি।  
আমি মূল লেখায়/কোডে ব্যাখ্যাটুকু যোগ করে দিচ্ছি,ধন্যবাদ।

[sumit93](http://uhunt.felix-halim.net/id/155831)

int ret=-inf; ei line ta bujlam na.-inf diye ki mean kora hocca??

শাফায়েত

-inf মানে হলো negative infinity,মানে খুব ছোট একটা সংখ্যা,যেমন -1000000000।

[sumit93](http://uhunt.felix-halim.net/id/155831)

উপরের অ্যারেতে সর্বোচ্চ পয়েন্ট ৭(-১+-২+১০) ai line e problem asa.Er theke to boro value kora jai.

শাফায়েত

কিভাবে? মুভ কিন্তু শুধু ৩দিকে করা যাবে,সরাসরি নিচে বা নিচের দুই কোনায়।

[sumit93](http://uhunt.felix-halim.net/id/155831)

Thanks vai vul ta amari   
ami vlo kore pori nai

মুয়ায

ভাই light oj 1004 problem আপনার মত করে করলাম ।কিন্তু wa কেন । একটু দেখবেন ।

[my code light oj 1004](http://pastebin.com/xrAtB2RC)

শুভ

<http://codepad.org/clATGtJk>  
একটু দেইখেন ভাইয়া :(

Rony

Sir, you have done a great job. first of all i would like to thank you for such a good work.if you kindly check my code and let me know where i did mistake , it will be great help to me.  
Problem Name : supper sale  
here is my code:

#include  
#include  
using namespace std;

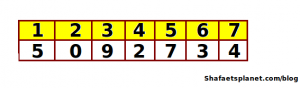
int dp[1010][105];  
int w[1010],cost[1010],N;  
int cap[110],g,capa;  
int f(int i,int W){  
if(i==N+1)  
return 0;  
if(dp[i][W]!=-1)  
return dp[i][W];  
int p1=0,p2=0;  
if(W+w[i]>=capa)  
p1=cost[i]+f(i+1,W+w[i]);  
p2=f(i+1,W);  
dp[i][W]=max(p1,p2);  
return dp[i][W];  
}  
int main(){  
int t;  
cin>>t;  
while(t–){

cin>>N;  
int r=0;  
for(int i=1;i>cost[i]>>w[i];  
cin>>g;  
for(int i=1;i>cap[i];  
for(int i=1;i<=g;i++){  
capa=cap[i];  
memset(dp,-1,sizeof dp);  
r+=f(1,0);  
}

cout<<dp[0][1]<<endl;  
cout<<r<<endl;  
}  
return 0;  
}

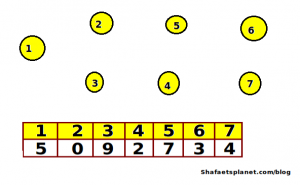
[**ডাইনামিক প্রোগ্রামিং এ হাতেখড়ি-৪**](http://www.shafaetsplanet.com/planetcoding/?p=1211)

[০-১ ন্যাপস্যাক,কয়েন চেঞ্জ](http://www.shafaetsplanet.com/planetcoding/?tag=%E0%A6%A1%E0%A6%BF%E0%A6%AA%E0%A6%BF) 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। এছাড়া দেখবো বুলিয়ান অ্যারে ব্যবহার করে ডিপি অপটিমাইজেশনের ছোট একটা ট্রিকস।

প্রথমেই শুরু করি LIS এবং এটার সলিউশন প্রিন্ট করা দিয়ে। LIS হলো Longest increasing subsequence। মনে করো তোমাকে একটি অ্যারে বা sequence দেয়া আছে:[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2012/08/lis1.png)  
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  
0 2 3 4  
7 3  
5 0 9 2 7 3 4 (শুন্যটি সংখ্যা বাদ দেয়া হয়েছে)  
….

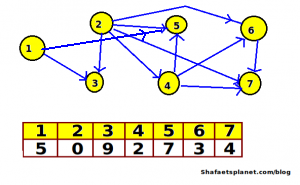
একটা sequence এ n টি সংখ্যা থাকলে subsequence থাকতে পারে মোট 2^n টি(কেন??)। increasing subsequence**(IS)** এ প্রতিটা সংখ্যা তার আগের সংখ্যাটির থেকে বড় হবে। উপরে ১ম , ৩য়, ৪র্থ subsequence গুলো increasing। যতগুলো IS আছে তারমধ্যে সবথেকে বড়টা খুজে বের করাই আমাদের লক্ষ্য। উপরে ৪নম্বরটির length ৪ এবং সবগুলো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

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2012/08/lis2.png)

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:

[](http://www.shafaetsplanet.com/planetcoding/?attachment_id=1483)

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];

}

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](http://www.lightoj.com/article_show.php?article=1000) 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](http://en.wikipedia.org/wiki/Directed_acyclic_graph) বা dag। কোনো প্রবলেমকে তুমি যদি dag এ কনভার্ট করতে পারো তাহলে খুব ভালো সম্ভাবনা আছে যে প্রবলেমটি ডিপি চালিয়ে সলভ করা যাবে। গ্রাফে cycle থাকলে longest path একটি np-complete প্রবলেম,অর্থাত polynomial টাইমে সলভ করা সম্ভবনা। 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

.........

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।

[](http://www.shafaetsplanet.com/planetcoding/?attachment_id=1520)

সবশেষে বুলিয়ান ট্রিকস। আমরা সাধারণত এভাবে লিখি:

if(dp[i][j][k]==-1) return dp[i][j][k];  
do rest of the work  
dp[i][j][k]=ans;  
return dp[i][j][k];

প্রতিবার ফাংশন কল করার সময় আমরা memset ব্যবহার করে ডিপি array তে -১ ভ্যালু বসাই। অ্যারের সাইজ বড় হলে বারবার এভাবে মেমসেট করা সময়সাপেক্ষ। integer এর থেকে boolean অ্যারে memset করতে সময় কম লাগে। তাই আমরা একটা বুলিয়ান vis অ্যারে রেখে লিখতে পারি:

if(vis[i][j][k]==true) return dp[i][j][k];  
do rest of the work  
vis[i][j][k]=true;dp[i][j][k]=ans;  
return dp[i][j][k];

ডিপি অ্যারের ১০০০ বা তার থেকেও বড় হলে এবং টেস্ট কেস অনেক থাকলে এভাবে অনেক সময় বাচানো যায় যেটা tle কোডকে accepted করে দিতে পারে।

আজকের পর্ব এখানেই শেষ। light oj 1231 এর সলিউশন: <http://pastebin.com/vmpWp0g1>, mod করার অংশটা বুঝতে না পারলে [এই লেখাটা দেখো](http://www.shafaetsplanet.com/planetcoding/?p=936)। সলভ করার জন্য প্রবলেম:

[Diving for gold](http://uva.onlinejudge.org/external/9/990.html) (uva-990)  
[Sense of Beauty](http://acm.timus.ru/problem.aspx?space=1&num=1501)  
[Bicolored Horses](http://acm.timus.ru/problem.aspx?space=1&num=1167)  
[Testing the CATCHER (uva- 231)](http://uva.onlinejudge.org/external/2/231.html)   
[How Many Dependencies?](http://uva.onlinejudge.org/external/109/10926.html)(uva- 10926)  
[Longest Path](http://uva.onlinejudge.org/external/100/10000.html) (uva-10000)

**মন্তব্য**

মুকিত

Lightoj 1231 নাম্বার নিয়ে একটা প্রশ্ন আছে…  
এর আগের পোস্ট থেকে জেনেছিলাম,একই আইটেম একাধিকবার নিলে জাস্ট i+1 এর পরিবর্তে i লিখলেই চলবে…কিন্তু ১২৩১ নং প্রবলেম করতে গিয়ে এটা মানা হয়নি…এটাতে আমি যখন coin[i] নিয়ে make বানানোর try করছি,তখনও i+1 দিয়ে call করছি ( এইক্ষেত্রে শুধু i লিখলে হচ্ছে না কেন ),আবার যখন coin[i] না নিয়ে try করছি,তখনও i+1 call করছি…  
উত্তরটা পেলে আরো উপকৃত হব …

শাফায়েত

এখানে ব্যাপারটা একটু ভিন্ন। ভিতরের for লুপটা দিয়ে আমরা একটা আইটেম কয়বার নিবো সেটা বলে দিচ্ছি,তাই আবার i কল করা দরকার নাই। কোনো আইটেম যদি লিমিটেড সংখ্যকবার নেয়া যায় তাহলে ভিতরে লুপ চালিয়ে কয়বার নিচ্ছি সেটা বলে দেয়া যায়। আগেরবার প্রতিবার ১টা করে নিচ্ছিলাম,তাই একইটা আরেকবার নিতে চাইলে i তে কল দেয়া দরকার ছিলো।

মুকিত

আরেকটি প্রশ্ন…  
profit1 যখন profit2 থেকে বড় হচ্ছে তখন আমি dir[] array তে 1 রাখছি(অর্থাৎ সেটি প্রিন্ট করব)…কিন্তু যখন equal হচ্ছে তখন নিচ্ছি না কেন? logic টা বললে আরো ভাল বুঝতে পারতাম… এই ব্যাপারটিতে problem হচ্ছে…

শাফায়েত

আসলে profit1==profit2 হলে dir অ্যারেতে ১ বা ২ কিছু একটা রাখলেই হবে, একটা ন্যাপস্যাক প্রবলেমের অনেকগুলো সলিউশন থাকতে পারে।

স্বপ্নবাজ

n element এর একটা array দেয়া আছে, সেটা থেকে exactly k সংখ্যক সংখ্যা নিতে হবে যাতে তাদের যোগফল w এর সবচেয়ে কাছাকাছি এবং <=w হয়। এই problemটা O(nkw)এর চেয়ে efficiently কিভাবে করা যাবে???

[**ডাইনামিক প্রোগ্রামিং এ হাতেখড়ি-৫**](http://www.shafaetsplanet.com/planetcoding/?p=1357)

আশা করি তুমি এখন [lis,knapsack,coin-change](http://www.shafaetsplanet.com/planetcoding/?tag=%E0%A6%A1%E0%A6%BE%E0%A6%87%E0%A6%A8%E0%A6%BE%E0%A6%AE%E0%A6%BF%E0%A6%95-%E0%A6%AA%E0%A7%8D%E0%A6%B0%E0%A7%87%E0%A6%BE%E0%A6%97%E0%A7%8D%E0%A6%B0%E0%A6%BE%E0%A6%AE%E0%A6%BF%E0%A6%82) প্রবলেম সলভ করতে পারো খুব সহজেই, ডিপির সলিউশন প্রিন্ট করতেও তোমার সমস্যা হয়না। এখন আমরা একটু অন্যরকম ডিপি দেখবো যেটার নাম বিটমাস্ক ডিপি। নামটা শুনে ভয় লাগলেও জিনিসটা সহজ, অনেক ক্ষেত্রেই বিটমাস্ক ডিপি প্রবলেম পড়ার সাথে সাথে সলিউশন মাথায় চলে আসে। তবে এই পর্বটা পড়ার আগে তোমাকে বিট নিয়ে কাজ করা শিখতে হবে, যেমন কোনো নির্দিষ্ট পজিশনের বিট অন করা/অফ করা ইত্যাদি। এজন্য তুমি এই চমৎকার [টিউটোরিয়ালটা](http://zobayer.blogspot.com/2009/12/bitwise-operations-in-cc-part-1.html) দেখতে পারো, পুরোটা খুবই ভালো করে পড়বে কারণ এটা তোমাদের অনেক জায়গায় কাজে লাগবে। আমি এই টিউটোরিয়ালে বিট অপারেশন নিয়ে লিখছিনা কারণ অপ্রাসঙ্গিক হয়ে যাবে।

আমরা শুরুতেই ৩টি ফাংশন ডিফাইন করি।

int Set(int N,int pos){return N=N | (1<<pos);}

int reset(int N,int pos){return N= N & ~(1<<pos);}

bool check(int N,int pos){return (bool)(N & (1<<pos));}

Set ফাংশনটি N সংখ্যাটির pos তম পজিশনের বিট ১ করে দেয়, reset ফাংশনটি 0 করে দেয় এবং check ফাংশনটি pos তম বিটে কি আছে সেটা রিটার্ণ করে। যেকোনো বিটমাস্ক ডিপি প্রবলেমে ফাংশন ৩টি দরকার হবে, বিশেষ করে Set এবং check।

আমরা একটা প্রবলেম দিয়ে শুরু করি। মনে করো তোমাকে nটা দোকান থেকে n টা জিনিস কিনতে হবে। জিনিসগুলো কিনতে তোমার a0,a1,a2,…,a(n-1) টাকা লাগে। তোমার শহরটা খুব অদ্ভূত,তুমি যখন একটা জিনিস কিনে আরেক দোকানে যাও তখন সেই দোকানদার তোমার আগের কেনা জিনিসগুলো দেখে তার দোকানের জিনিসের দাম বাড়িয়ে দেয়!! কত দাম বাড়াবে সেটা নির্ভর করবে তুমি আগে আগে কোন কোন দোকানে গিয়েছো সেটার উপর। ধরো n=2, তাহলে তোমাকে নিচের মতো একটা ম্যাট্রিক্স দেয়া থাকবে:

10 10  
90 10

এখন,

যদি (i==j) হয় তাহলে matrix[i][j]=matrix[i][i]=i’তম জিনিসটির আসল দাম।  
যদি (i!=j) হয় তাহলে matrix[i][j]=j’তম জিনিসটি আগে কিনলে i’তম জিনিসটির সাথে যোগ হওয়া বাড়তি দাম।

তুমি যদি শুরুতে ০ তম জিনিসটা কিনো তাহলে দাম পড়বে ১০টাকা, এরপর ১নম্বর জিনিসটা কিনলে সেটার দাম হবে ১০+৯০ টাকা, কারণ matrix[1][0]=০ নম্বর জিনিসের আগে ১ নম্বর জিনিস কিনলে যোগ হওয়া বাড়তি দাম=৯০ আর ১ নম্বর জিনিসের আসল দাম=১০, তাহলে মোট খরচ ১০+(১০+৯০)=১১০। কিন্তু তুমি যদি ১নম্বর জিনিসটা আগে কিনো তাহলে মোট খরচ ১০+(১০+১০)=৩০ টাকা।  
বুঝতেই পারছো তোমার কাজ হলো খরচ মিনিমাইজ করা। n এর মান সর্বোচ্চ ১৫।

n এর মান খুব কম বলে বিটমাস্ক ডিপি দিয়ে প্রবলেমটি সহজেই সলভ করা যাবে। ডিপিতে আমাদের প্রথম কাজ হলো স্টেট নির্ণয় করা। এই কেনাকাটার যেকোনো সময় আমাদের অবস্থা কি কি তথ্য দিয়ে প্রকাশ করা যায়? “এখন পর্যন্ত কোন কোন জিনিস কেনা হয়েছে” এই তথ্যটাই যথেষ্ট, তাইনা? এটা জানলে আমরা পরবর্তি আরেকটি জিনিস কেনার সময় বাড়তি কত খরচ যোগ হবে জানতে পারবো, পরবর্তিতে যেই জিনিসটা কিনলে মোট খরচ কম হবে সেটা আমরা কিনবো। মনে করি এখন কথা হলো স্টেটটা রাখবো কি ভাবে?

একটা উপায় হলো n টি জিনিসের জন্য এভাবে nটা স্টেট রাখা function(a0,a1,a1,….an-1), কিন্তু n এর মান বদলালে তুমি প্যারামিটার সংখ্যা বদলাবে কি ভাবে? আর ১৫টি প্যারামিটার নিয়ে কাজ করলে কোডটা ভয়াবহ জটিল হয়ে যাবে।

২য় উপায় হলো বিটমাস্ক। একটি ইন্টিজারে ৩২টি বিট থাকে। আমরা সেই সুবিধাটাই নিবো। ১ নম্বর বিট 1 হলে আমরা ১ নম্বর জিনিসটা নিয়েছি, ০ হলে নেইনি। ৩ নম্বর বিট 1 হলে আমরা ৩ নম্বর জিনিসটা নিয়েছি, ০ হলে নেইনি। ১ এবং ৩ নম্বর বিট দুইটাই 1 হলে আমরা ২টা জিনিসই নিয়েছি।

শুরুতে আমাদের স্টেট থাকবে 0 বা বাইনারিতে “000000″। তারমানে আমরা কোনো জিনিস এখনও কিনিনি। ০তম এবং ১তম জিনিস কেনা হয়ে গেলে স্টেট হবে 3 বা “000011″ এবং n=2 এর জন্য এটাই আমাদের base case। n=4 এর জন্য base case হলো 15 বা “0001111″। leading zero নিয়ে চিন্তা করা দরকার নেই, এটা বোঝার সুবিধার্থে দেয়া হয়েছে। একটু চিন্তা করলেই বুঝতে পারবে mask=(2^n)-1 হলে সেটা হবে base case কারণ তখন বাইনারিতে প্রথম n টা বিট ১ থাকবে, আমরা তখন শুন্য রিটার্ণ করে দিবো তখন কারণ আর কোনো জিনিস কেনা বাকি নেই।

int dp[(1<<15)+2];

int call(int mask)

{

if(mask==(1<<n)-1) return 0;

if(dp[mask]!=-1) return dp[mask];

//Rest of the calculation

}

বেসকেস বুঝলাম, এরপরে আমাদের কাজ হবে যেসব জিনিস কেনা হয়নি সেগুলা নিয়ে চেষ্টা করে দেখা। mask এর i তম পজিশনে যদি ০ থাকে তাহলে i তম জিনিসটি কেনা এখনও বাকি আছে।

int dp[(1<<15)+2];

int call(int mask)

{

if(mask==(1<<n)-1) return 0;

if(dp[mask]!=-1) return dp[mask];

//Rest of the calculation

int ans=1<<28; //Infinite, a large value

for(int i=0;i<n;i++)

{

if(check(mask,i)==0)

{

//Rest of the code

}

}

return dp[mask]=ans;

}

আমরা n পর্যন্ত লুপ চালিয়ে বের করে নিলাম কোনটা কোনটা নেয়া বাকি আছে। এখন i তম জিনিসটার আসল দাম হলো price=w[i][i]। এই price এর সাথে w[i][j] যোগ হবে যদি i!=j হয় এবং j নম্বর জিনিসটা আগেই কেনা হয়ে থাকে। mask এর j তম বিট চেক করে আমরা বলতে পারি j তম জিনিসটা কেনা হয়েছে নাকি।

int dp[(1<<14)+2];

int call(int mask)

{

if(mask==(1<<n)-1) return 0;

if(dp[mask]!=-1) return dp[mask];

int ans=1<<28;

for(int i=0;i<n;i++)

{

if(check(mask,i)==0)

{

int price=w[i][i];

for(int j=0;j<n;j++)

if(i!=j and check(mask,j)!=0) price+=w[i][j];

int ret=price+call(Set(mask,i));

ans=min(ans,ret);

}

}

return dp[mask]=ans;

}

j এর লুপটা দিয়ে আমরা মোট দাম বের করে নিলাম। এখন i তম জিনিসটি কিনলে পরবর্তি স্টেট কি হবে? শুধু mask এর i তম বিটটি 1 করে দিতে হবে। আমরা call(Set(mask,i)) এভাবে i তম জিনিস কিনে পরবর্তি স্টেটে চলে গেলাম। এভাবে প্রতিটি জিনিস কিনে যেটায় দাম মিনিমাম হয় সেটা রিটার্ণ করে দিলাম। কাজ শেষ! সম্পূর্ণ কোড:

int w[20][20];

int n;

int dp[(1<<15)+2];

int call(int mask)

{

if(mask==(1<<n)-1) return 0;

if(dp[mask]!=-1) return dp[mask];

int mn=1<<28;

for(int i=0;i<n;i++)

{

if(check(mask,i)==0)

{

int price=w[i][i];

for(int j=0;j<n;j++)

{

if(i!=j and check(mask,j)!=0)

{

price+=w[i][j];

}

}

int ret=price+call(Set(mask,i));

mn=min(mn,ret);

}

}

return dp[mask]=mn;

}

int main()

{

mem(dp,-1);

cin>>n;

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

scanf("%d",&w[i][j]);

}

}

int ret=call(0);

printf("%d\n",ret);

return 0;

}

আমাদের call ফাংশনটি কয়টি ভিন্ন স্টেটে থাকতে পারে? n টি বিটের প্রতিটি হয় ০ হবে নাহয় ১ হবে, তাহলে স্টেট থাকতে পারে 2^15 টি। আর ভিতরে একটা n^2 লুপ চলছে তাই মোট complexity (2^n)\*(n^2)।

বিটমাস্ক ডিপি চেনার সবথেকে সহজ উপায় n এর মান দেখা। n এর মান ১৬ বা তার কম হলে খুব ভালো সম্ভাবনা আছে যে প্রবলেমটিকে বিটমাস্ক ডিপি দিয়ে সলভ করা যাবে।  
বিটমাস্ক ডিপি কখন ব্যবহার করবো? বিটমাস্ক লাগবে আমাদের তখনই যখন আগের স্টেটে কোন কোন জিনিস/ডিজিট/নোড ইত্যাদি ব্যবহার করা হয়েছে সে তথ্যটি আমার বর্তমান স্টেটে লাগবে। সেই তথ্য অনুযায়ী আমরা বর্তমান স্টেট থেকে নতুন ডিজিট/নোড ইত্যাদি নিবো এবং সেই বিটটি অন করে দিয়ে সামনের স্টেটে যাবো। যখন n টি বিট অন হয়ে যাবে তখন বেসকেস রিটার্ণ করে দিবো।

আমার সলভ করা প্রথম বিটমাস্ক ডিপি হলো [uva 10651](http://uva.onlinejudge.org/external/106/10651.html)। আশা করি প্রবলেমটি এখন সহজেই করতে পারবে। প্রবলেমটায় একটি মাস্কের সাহায্যে কোনো সময় বোর্ডের কি অবস্থা সেই তথ্যটা রাখবে, এবং সে অবস্থায় যতগুলো চাল দেয়া সম্ভব সবগুলো দিয়ে মিনিমামটা রিটার্ণ করে দিবে।

আপাতত এগুলোই ছিলো বিটমাস্ক ডিপির বেসিক। সামনের পর্বগুলোতে আরো বিস্তারিত আলোচনার চেষ্টা করবো। এখন নিচের প্রবলেমগুলো সলভ করার চেষ্টা করো, ১ম প্রবলেমটি নিয়ে এই পর্বে আলোচনা করেছি:  
[Pimp My Ride](http://www.lightoj.com/volume_showproblem.php?problem=1119) (loj-1119)  
[False Mirror](http://acm.timus.ru/problem.aspx?space=1&num=1152)  
[Agent 47](http://www.lightoj.com/volume_showproblem.php?problem=1037) (loj - 1037)  
[Painful Bases](http://www.lightoj.com/volume_showproblem.php?problem=1021) (loj - 1021)

[আবু আসিফ খান চৌধুরী](http://abuasifkhan.blogspot.com)

বরাবরের মত অনেক সুন্দর হইছে লেখাটা। আপনার লেখাগুলার জন্য অপেক্ষা করে থাকি। তবে ভাইয়া, বিটমাস্ক ডিপি আমরা কেন ব্যবহার করবো একটু বললে ভাল হত। মেমরী বাচানো ছাড়া আর সুবিধাগুলা কি? আর বিটমাস্কিং ছাড়া কি প্রবলেমটা অন্যকোন ভাবে সল্ভ করা যেত না?

শাফায়েত

ভালো প্রশ্ন। প্রবলেমটায় n টি জিনিসের জন্য এভাবে nটা স্টেট রাখা function(a0,a1,a1,….an-1) যায়, কিন্তু n এর মান বদলালে তুমি প্যারামিটার সংখ্যা বদলাবে কি ভাবে? আর ১৫টি প্যারামিটার নিয়ে কাজ করলে কোডটা ভয়াবহ জটিল হয়ে যাবে। বিটমাস্ক লাগবে আমাদের তখনই যখন আগের স্টেটে কোন কোন জিনিস/ডিজিট/নোড ইত্যাদি ব্যবহার করা হয়েছে সে তথ্যটি আমার বর্তমান স্টেটে লাগবে। বিটমাস্ক ছাড়া একটা মাত্র ভ্যারিয়েবলের সাহায্যে ইনফরমেশনগুলা পাস করা যাবেনা।  
মূল লেখায় কথাগুলো যোগ করে দিলাম।

রূপক

মেমরী বাচানো ছাড়া আর সুবিধাগুলা কি? – Bitwise অপারেশন এর এক্সিকিউসন টাইম অনেক কম এবং অনেক বেশি মেমরি এফিসিয়েন্ট। তাই যদিও function(a0,a1,a1,….an-1) এর সমস্যাগুলো function(a[]) (প্যারামিটার a একটি ইন্টিজার Array) দ্বারা সমাধান করা সম্ভব, কিন্তু এটি অনেক ভালো অ্যালগোরিদম-এর পারফরমেন্সও অনেক বেশি কমিয়ে আনবে। @শাফায়েত – অনেক ভালো হয়েছে টিউটোরিয়ালটি।

শাফায়েত

ডিপিতে আমরা প্রতিটা স্টেট একটা টেবিলে সেভ করে রাখি, অ্যারে টেবিলে সেভ করবে কিভাবে? ম্যাপিং করা যেতে পারে কিন্তু তাহলে প্রতিবার টেবিল access করতে logn টাইম লাগবে, কোড নিশ্চিতভাবে tle দিবে।  
ধন্যবাদ।

রূপক

টেবিল (matrix) একটা Data Structure যেটা আমাদেরকে আগের সেভ করা তথ্য গুলো দ্রুত খুঁজে পেতে (Lookup) সাহায্য করবে – তাই না? টেবিল যে সবসময় একটা 2D Integer Array হতে হবে এমন তো না – আমার ইমপ্লিমেন্টেশন এ হয়ত আমি matrix Array টার ডাইমেনশন বাড়িয়ে নিতে পারি (Java-তে একটা Array কিন্তু একটা Object)। আমি আসলে শুধুমাত্র এই লাইন এর জন্য, “বিটমাস্ক ছাড়া একটা মাত্র ভ্যারিয়েবলের সাহায্যে ইনফরমেশনগুলা পাস করা যাবেনা” আমার আগের কমেন্টটা করেছিলাম। ধন্যবাদ।

শাফায়েত

আসলে একটি ভ্যারিয়েবল বলতে আমি একটি single integer বুঝিয়েছিলাম, array তো একটা collection of variable তাইনা? জাভাতে আপনি একটি অ্যারেকে টেবিল থেকে O(1) এ খুজে বের করতে পারবেন? আমার জানার ভুল হতে পারে তবে আমার মনে হয় পারবেননা, অবজেক্টকে টেবিলে রাখতে হলে হ্যাশটেবিলে রাখতে হবে, যেখান থেকে খুজতে logn সময় লাগে। সি তেও আপনি একটা ক্লাস বা স্ট্রাকচার অ্যারে রেখে সেটাকে অবজেক্ট বানিয়ে টেবিলে রাখতে পারেন, তখনও হ্যাশটেবিল ব্যবহার করতে হবে।  
ভুল বললে ধরিয়ে দিতে পারেন, আলোচনা সবসময়ই শিখতে সাহায্য করে :)।

এডিট: জাভাতে হ্যাশম্যাপ থেকে অ্যাভারেজে O(1) সময়ে অবজেক্ট খুজে পাওয়া সম্ভব। সি++ এ ম্যাপে bst ব্যবহার করা যেটায় logn এ সার্চ করতে হয় কিন্তু জাভার hashtable এ অ্যাভারেজে O(1) এ সার্চ করে তবে worst case এ সেটা O(n) এ চলে যায়।

রূপক

Don’t Get Me Wrong :)– আমি আসলে ভুল ধরিয়ে দিতে চাচ্ছিনা। একটা Function এ Parameter পাঠানোর সমস্যার চেয়ে Memory এবং Runtime Efficiency কে আমার কাছে গুরুত্বপূর্ণ মনে হয়েছে। কারণ প্রথম সমস্যাটা অনেক বেশি Language Specific.  
আর Java – Array – Object নিয়ে আমি যা বলতে চেয়েছি সেটা নিচের লিংক এক্সপ্লোর করলে জানা যাবে (জাভাতে আপনি একটি অ্যারেকে টেবিল থেকে O(1) এ খুজে বের করতে পারবেন? – সম্ভব) -  
১। [Java Language Specification – Chapter 10. Arrays.](http://docs.oracle.com/javase/specs/jls/se7/html/jls-10.html)  
২। [Is An Array an Object in Java.](http://stackoverflow.com/questions/8781022/is-an-array-an-object-in-java)  
ধন্যবাদ :)

শাফায়েত

O(1) এ খুজে বের করা যাবে এই কথা specifically কোথায় লেখা আছে বলতে পারবেন? অ্যারে অবজেক্ট সেটা অবশ্যই ঠিক আছে, একটা অবজেক্টকে identify করতে হলে একটা হ্যাশভ্যালু ক্যালকুলেট করা দরকার, তারপর সেটাকে হ্যাশটেবিল থেকে বের করতে হবে। আপনি একটু specific ভাবে দেখান কোথায় লেখা আছে সেটা, লিংকগুলোতে আমি তেমন কিছু পাইনি। অথবা আপনি সেরকম কোনো কোড পোস্ট করতে পারেন|

শাফায়েত

আমি নেটে সার্চ করলাম একটু, কিছু নতুন তথ্য জানলাম যেগুলো জানা ছিলোনা। “Hash tables are O(1) average and amortized case complexity, however is suffers from O(n) worst case time complexity. ” এরকম লেখা পেলাম। আমি ধারণা করেছিলাম জাভার হ্যাশটেবিল সি++ এর ম্যাপের মতোই logn এ সার্চ করে। তারমানে জাভাতে অবজেক্ট average কেসে কেস এ O(1) এ খুজে পাওয়া সম্ভব। ধন্যবাদ আপনাকে।

রূপক

Check this code, <http://pastebin.com/2pAwtuJx>, specially line number 19 and 23/27 where previously stored values were being retrieved.