**Binomial Coefficient**

**int sieve(int n,int r){**

**if(r==0||r==n) return 1;**

**if(dp[n][r]!=-1) return dp[n][r];**

**else{**

**dp[n][r]=sieve(n-1,r-1)+sieve(n-1,r);**

**return dp[n][r];**

**}**

**}**

**0-1 Knapsack**

int n,w;

int weight[mx],price[mx],dp[mx][mx];

int sieve(int x,int y){

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

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

else{

int ret1,ret2;

if(y+weight[x]<=w) ret1=sieve(x+1,y+weight[x])+price[x];

else ret1=0;

ret2=sieve(x+1,y);

dp[x][y]=max(ret1,ret2);

return dp[x][y];

}

}

int main(){

int t;

scanf("%d",&t);

for(int tc=1;tc<=t;tc++){

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

memset(weight,0,sizeof(weight));

memset(price,0,sizeof(price));

scanf("%d %d",&n,&w);

for(int i=1;i<=n;i++){

int x,y; scanf("%d %d",&x,&y);

weight[i]=x; price[i]=y;

}

int ans=sieve(1,0);

printf("Case %d: %d\n",tc,ans);

}

return 0;

}

Input: 5 10

১. মানিব্যাগ: ১ পাউন্ড, ১২০টাকা  
২. কোরম্যানের-বই: ৭ পাউন্ড, ৪০০টাকা  
৩. ডিভিডি-কালেকশন: ৪ পাউন্ড, ২৮০ টাকা  
৪. ফেলুদা-সমগ্র: ৩ পাউন্ড, ১৫০টাকা  
৫. ফুটবল: ভর: ৪ পাউন্ড, ২০০টাকা Output: Answer : 600 TK.

**Coin Change**

**Using Namespace std;**

**int dp[6][7590],price[]={50,25,10,5,1};**

**int make;**

|  |  |
| --- | --- |
| **Time limit** | **Optimization** |
| int sieve(int x,int y){  if(x>=5){  if(y==make) return 1;  else return 0;  }  if(dp[x][y]!=-1) return dp[x][y];  else{  int ret1,ret2;  if(y+price[x]<=make) ret1=sieve(x,y+price[x]);  else ret1=0;  ret2=sieve(x+1,y);  dp[x][y]=ret1+ret2;  return dp[x][y];  }  }  int main(){  while(scanf("%d",&make)==1){  memset(dp,-1,sizeof(dp));  int ans=sieve(0,0);  printf("%d\n",ans);  }  return 0;  } | int sieve(int x,int y){  if(x>=5){  if(y==0) return 1;  else return 0;  }  if(dp[x][y]!=-1) return dp[x][y];  else{  int ret1,ret2;  if(y-price[x>=0) ret1=sieve(x,y-price[x]);  else ret1=0;  ret2=sieve(x+1,y);  dp[x][y]=ret1+ret2;  return dp[x][y];  }  }  int main(){  memset(dp,-1,sizeof(dp));  while(scanf("%d",&make)==1){  int ans=sieve(0,make);  printf("%d\n",ans);  }  return 0;  } |

**If coin is limited:**

using namespace std;

long long int dp[51][1001];

int price[51],times[51], n, k;

|  |  |
| --- | --- |
| int main(){  int t;  scanf("%d",&t);  for(int tc=1;tc<=t;tc++){  scanf("%d %d",&n,&k);  memset(dp,-1,sizeof(dp));  memset(price,0,sizeof(price));  memset(times,0,sizeof(times));  for(int i=1;i<=n;i++) scanf("%d",&price[i]);  for(int i=1;i<=n;i++) scanf("%d",&times[i]);  long long int ans=sieve(1,0)%mod;  printf("Case %d: %lld\n",tc,ans);  }  return 0;  } | long long int sieve(int x,int y){  if(x==n+1){ if(y==k) return 1;  else return 0;  }  //if(y==k) return 1;  if(dp[x][y]!=-1) return dp[x][y];  else{ int ret1=0,ret2;  for(int i=1;i<=times[x];i++){  if(y+price[x]\*i<=k) ret1+=sieve(x+1,y+price[x]\*i);  else ret1+=0;  }  ret2=sieve(x+1,y);  dp[x][y]=(ret1+ret2)%mod;  return dp[x][y]%mod;  }  } |

**Rock Climbing**

using namespace std;

map<int,vector<int> >arr;

int n,dp[101][101];

|  |  |
| --- | --- |
| int main(){  int t; scanf("%d",&t);  for(int tc=1;tc<=t;tc++){  scanf("%d",&n);  for(int i=0;i<n;i++) for(int j=1;j<=n;j++){  int k; scanf("%d",&k); arr[i].push\_back(k);  }  memset(dp,-1,sizeof(dp));  int ans=sieve(0,0);  printf("Case %d: %d\n",tc,ans);  for(int i=1;i<=n;i++) arr[i].clear();  }  return 0;  } | int sieve(int tx,int ty){  if(valid(tx,ty)){  if(dp[tx][ty]!=-1) return dp[tx][ty];  int ret=0;  for(int i=0;i<3;i++){  ret=max(ret,sieve(tx+fx[i],ty+fy[i])+arr[tx][ty]);  }  dp[tx][ty]=ret;  return dp[tx][ty];  }  else return 0;  } |

**Dp Solution Print**

int n,w;

int weight[mx],price[mx],dp[mx][mx],dir[mx][mx];

|  |  |
| --- | --- |
| int main(){  int t;  scanf("%d",&t);  for(int tc=1;tc<=t;tc++){  memset(dp,-1,sizeof(dp));  memset(dir,-1,sizeof(dir));  memset(weight,0,sizeof(weight));  memset(price,0,sizeof(price));  scanf("%d %d",&n,&w);  for(int i=1;i<=n;i++){  int x,y;  scanf("%d %d",&x,&y);  weight[i]=x;  price[i]=y;  }  int ans=sieve(1,0);  printf("Case %d: %d\n",tc,ans);  int total=print(1,0);  printf("Total things %d\n",total);  for(int i=0;i<sol.size();i++)  printf("%d %d\n",weight[sol[i]],price[sol[i]]);  sol.clear();  }  return 0;  } | int sieve(int x,int y){  if(x==n+1) return 0;  if(dp[x][y]!=-1) return dp[x][y];  else{  int ret1,ret2;  if(y+weight[x]<=w)  ret1=sieve(x+1,y+weight[x])+price[x];  else ret1=0;  ret2=sieve(x+1,y);  if(ret1>ret2){ dir[x][y]=1;  return dp[x][y]=ret1;  }  else{ dir[x][y]=2;  return dp[x][y]=ret2;  }  }  }  vector<int>sol;  int print(int x,int y){  if(dir[x][y]==-1) return 0;  if(dir[x][y]==1){  sol.push\_back(x);  return 1+print(x+1,y+weight[x]);  }  else return print(x+1,y);  } |

**UVA – 990 Diving for Gold**

int t,w,n,depth[50],coin[50],dp[50][1001],dir[50][1001];

vector<int>sol;

|  |  |
| --- | --- |
| int main(){  bool flag=false;  while(scanf("%d %d",&t,&w)==2){  if(flag==true) printf("\n");  flag=true;  memset(dp,-1,sizeof(dp));  memset(dir,-1,sizeof(dir));  scanf("%d",&n);  for(int i=1;i<=n;i++)  scanf("%d %d",&depth[i],&coin[i]);  int ans=sieve(1,0);  printf("%d\n",ans);  int total=sol\_print(1,0);  printf("%d\n",total);  for(int i=0;i<sol.size();i++){  printf("%d %d\n",depth[sol[i]],coin[sol[i]]);  }  sol.clear();  }  return 0;  } | int sieve(int x,int y){  if(x==n+1) return 0;  if(dp[x][y]!=-1) return dp[x][y];  else{  int ret1,ret2;  if(y+3\*w\*depth[x]<=t)  ret1=sieve(x+1,y+3\*w\*depth[x])+coin[x];  else ret1=0;  ret2=sieve(x+1,y);  if(ret1>ret2){  dir[x][y]=1;  return dp[x][y]=ret1;  }  else{  dir[x][y]=2;  return dp[x][y]=ret2;  }  }  }  int sol\_print(int x,int y){  if(dir[x][y]==-1) return 0;  else if(dir[x][y]==1){  sol.push\_back(x);  return 1+sol\_print(x+1,y+3\*w\*depth[x]);  }  else sol\_print(x+1,y);  } |

**Longest Increasing Subsequence**

**An O(n2) approach**

using namespace std;

int m\_pos,n,arr[mx],dp[mx],sequence[mx];

int LIS(){

for(int i=0;i<mx;i++) dp[i]=1;

for(int i=0;i<n;i++) for(int j=i+1;j<n;j++)

if(arr[j]>arr[i]&&dp[j]<dp[i]+1) dp[j]=dp[i]+1;

int ans=0;

for(int i=0;i<n;i++) if(ans<dp[i]){ans=dp[i];m\_pos=i;}

return ans;

}

void lis\_print(int M){

int top=M-1;

sequence[top]=arr[m\_pos];top--;

for(int i=m\_pos-1;i>=0;i--) if(arr[i]<arr[m\_pos]&&dp[i]==dp[m\_pos]-1){

m\_pos=i;sequence[top]=arr[m\_pos];top--;

}

printf("LIS is :");

for(int i=0;i<M;i++) printf(" %d",sequence[i]);

printf("\n");

}

int main(){

scanf("%d",&n);

for(int i=0;i<n;i++) scanf("%d",&arr[i]);

int ans=LIS();

printf("The LIS length is %d\n",ans);

lis\_print(ans);

return 0;

}

**UVA - 231 Testing the CATCHER**

using namespace std;

int m\_pos,n,arr[mx],dp[mx],sequence[mx];

|  |  |
| --- | --- |
| int main(){  int t=0;  while(1){  n=1; int i=0;  scanf("%d",&arr[i++]);  if(arr[i-1]==-1) break;  while(1){  n++;  scanf("%d",&arr[i++]);  if(arr[i-1]==-1) break;  }  n--;  int ans=LIS();  if(t) printf("\n");  printf("Test #%d:\n",++t);  printf(" maximum possible interceptions: %d\n",ans);  }  return 0;  } | int LIS(){  for(int i=0;i<mx;i++) dp[i]=1;  for(int i=0;i<n;i++) for(int j=i+1;j<n;j++)  if(arr[j]<arr[i]&&dp[j]<dp[i]+1) dp[j]=dp[i]+1;  int ans=0;  for(int i=0;i<n;i++) if(ans<dp[i]) ans=dp[i];  return ans;  } |

**Longest Increasing Subsequence**

**An O(nlogk) Approach**

using namespace std;

int m\_pos,n,arr[mx],dp[mx],I[mx],sequence[mx];

|  |  |
| --- | --- |
| int main(){  scanf("%d",&n);  for(int i=0;i<n;i++) scanf("%d",&arr[i]);  int ans=LIS();  printf("The LIS length is %d\n",ans);  lis\_print(ans);  //lis\_print(ans);  return 0;  }  void lis\_print(int M){  int top=M-1;  sequence[top]=arr[m\_pos];top--;  for(int i=m\_pos-1;i>=0;i--) if(arr[i]<arr[m\_pos]&&dp[i]==dp[m\_pos]-1){  m\_pos=i;sequence[top]=arr[m\_pos];top--;  }  printf("LIS is :");  for(int i=0;i<M;i++) printf(" %d",sequence[i]);  printf("\n");  } | int LIS(){  I[0]=-inf;  for(int i=1;i<=n;i++) I[i]=inf;  int lis\_len=0;  for(int i=0;i<n;i++){  int low=0,high=lis\_len,mid;  while(low<=high){  mid=(low+high)/2;  if(I[mid]<arr[i]) low=mid+1;  else high=mid-1;  }  I[low]=arr[i];  dp[i]=low;  if(lis\_len<low){lis\_len=low;m\_pos=i;}  }  return lis\_len;  } |

**Bit Mask:- 1119 - Pimp My Ride-Light OJ**

**Input**

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with an integer **n (1 ≤ n ≤ 14)** denoting number of jobs. Then follow **n** lines, each containing exactly **n** integers. The **ith** line contains the surcharges that have to be paid in garage number **i** for the **ith** job and the base price for job **i**. More precisely, on the **ith** line, the **ith** integer is the base price for job **i** and the **jth**integer **i ≠ j** is the surcharge for job **i** that applies if job **j** has been done before. The prices will be non-negative integers smaller than or equal to **100000**.

**Output**

For each case, print the case number and the minimum total cost.

Solution:

int m;

int price[20][20];

int dp[(1<<15)+5];

int bit\_mask(int p){

if(p==(1<<m)-1) return 0;

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

int ret1=1<<28;

for(int i=0;i<m;i++){

if(!check(p,i)){

int w=price[i][i];

for(int j=0;j<m;j++) if(i!=j&&check(p,j)) w+=price[i][j];

int ret2=w+bit\_mask(biton(p,i));

ret1=min(ret1,ret2);

}

}

return dp[p]=ret1;

}

int main(){

int t;

scanf("%d",&t);

for(int cs=1;cs<=t;cs++){

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

scanf("%d",&m);

for(int i=0;i<m;i++) for(int j=0;j<m;j++) scanf("%d",&price[i][j]);

int ans=bit\_mask(0);

printf("Case %d: %d\n",cs,ans);

}

return 0;

}

Input: Output:

2

2 Case 1: 30

10 10

9000 10

3 Case 2: 42

14 23 0

0 14 0

1000 9500 14

**Bit Mask:- 1021 - Painful Bases**

**Input**

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. After that there will be two integers, **base (2 ≤ base ≤ 16)** and **K** **(1 ≤ K ≤ 20)**. The next line contains a valid integer in that base which contains distinct digits, that means in that number no digit occurs more than once.

**Output**

For each case, print the case number and the desired result.

Solution:

int arr[101];

ll dp[20][(1<<16)+5];

int base,k;

string s;

ll sieve(int value,int num){

ll &ret=dp[value][num];

if(one(num)==s.length()) return value==0; ; ///মানে সংখ্যাটা K দ্বারা বিভাজ্য কি না,হলে count ১ করে পাঠাবে...!

if(ret!=-1) return ret;

ret=0;

for(int i=0;i<s.length();i++) if(!bitcheck(num,i)) ret+=sieve((value\*base+arr[i])%k,biton(num,i));

return dp[value][num]=ret;

}

int main(){

int t;

scanf("%d",&t);

for(int cs=1;cs<=t;cs++){

scanf("%d %d",&base,&k);

s.clear();

cin>>s;

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

for(int i=0;i<s.length();i++){

if(s[i]>='0'&&s[i]<='9') arr[i]=s[i]-'0';

else arr[i]=s[i]-'A'+10;

}

ll ans=sieve(0,0);

printf("Case %d: %lld\n",cs,ans);

}

return 0;

}

Input:

2

10 2

5681

16 1

ABCDEF0123456789

Output:

Case 1: 12

Case 2: 20922789888000

# Printing Longest Common Subsequence

Given two sequences, print the longest subsequence present in both of them.

**Examples:**  
LCS for input Sequences “ABCDGH” and “AEDFHR” is “ADH” of length 3.  
LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length 4.

We have discussed [Longest Common Subsequence (LCS)](https://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence/) problem in a [previous post](https://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence/). The function discussed there was mainly to find the length of LCS. To find length of LCS, a 2D table L[][] was constructed. In this post, the function to construct and print LCS is discussed.

Following is detailed algorithm to print the LCS. It uses the same 2D table L[][].

**1)** Construct L[m+1][n+1] using the steps discussed in [previous post](https://www.geeksforgeeks.org/dynamic-programming-set-4-longest-common-subsequence).

**2)** The value L[m][n] contains length of LCS. Create a character array lcs[] of length equal to the length of lcs plus 1 (one extra to store \0).

**2)** Traverse the 2D array starting from L[m][n]. Do following for every cell L[i][j]  
…..**a)** If characters (in X and Y) corresponding to L[i][j] are same (Or X[i-1] == Y[j-1]), then include this character as part of LCS.  
…..**b)** Else compare values of L[i-1][j] and L[i][j-1] and go in direction of greater value.

The following table (taken from [Wiki](http://en.wikipedia.org/wiki/Longest_common_subsequence_problem)) shows steps (highlighted) followed by the above algorithm.

|  |
| --- |
| **/\* Dynamic Programming implementation of LCS problem \*/**  **#include<iostream>**  **#include<cstring>**  **#include<cstdlib>**  **using namespace std;**    **/\* Returns length of LCS for X[0..m-1], Y[0..n-1] \*/**  **void lcs( char \*X, char \*Y, int m, int n )**  **{**  **int L[m+1][n+1];**    **/\* Following steps build L[m+1][n+1] in bottom up fashion. Note**  **that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] \*/**  **for (int i=0; i<=m; i++)**  **{**  **for (int j=0; j<=n; j++)**  **{**  **if (i == 0 || j == 0)**  **L[i][j] = 0;**  **else if (X[i-1] == Y[j-1])**  **L[i][j] = L[i-1][j-1] + 1;**  **else**  **L[i][j] = max(L[i-1][j], L[i][j-1]);**  **}**  **}**    **// Following code is used to print LCS**  **int index = L[m][n];**    **// Create a character array to store the lcs string**  **char lcs[index+1];**  **lcs[index] = '\0'; // Set the terminating character**    **// Start from the right-most-bottom-most corner and**  **// one by one store characters in lcs[]**  **int i = m, j = n;**    **while (i > 0 && j > 0)**  **{**  **// If current character in X[] and Y are same, then**  **// current character is part of LCS**  **if (X[i-1] == Y[j-1])**  **{**  **lcs[index-1] = X[i-1]; // Put current character in result**  **i--; j--; index--;     // reduce values of i, j and index**  **}**    **// If not same, then find the larger of two and**  **// go in the direction of larger value**  **else if (L[i-1][j] > L[i][j-1])**  **i--;**  **else**  **j--;**  **}**    **// Print the lcs**  **cout << "LCS of " << X << " and " << Y << " is " << lcs;**  **}**    **/\* Driver program to test above function \*/**  **int main()**  **{**  **char X[] = "AGGTAB";**  **char Y[] = "GXTXAYB";**  **int m = strlen(X);**  **int n = strlen(Y);**  **lcs(X, Y, m, n);**  **return 0;**  **}** |
|  |

**Output:**

**LCS of AGGTAB and GXTXAYB is GTAB**

1157 – LCS Revisited

|  |  |
| --- | --- |
| int main(){  int t;  scanf("%d",&t);  for(int cs=1;cs<=t;cs++){  cin>>s1>>s2;  mem(aRight,-1);  mem(bRight,-1);  for(int i=0;i<=s1.size();i++)  for(int j=0;j<=s2.size();j++)  dp[i][j].first=-1;  for(int i=0;i<=s1.size();i++)  for(int j=0;j<=s2.size();j++)  dp[i][j].second=0;  for(int i=s1.size()-1;i>=0;i--){  for(int j=0;j<26;j++){  aRight[i][j]=aRight[i+1][j];  }  aRight[i][s1[i]-'a']=i;  }  for(int i=s2.size()-1;i>=0;i--){  for(int j=0;j<26;j++){  bRight[i][j]=bRight[i+1][j];  }  bRight[i][s2[i]-'a']=i;  }  int ans = func(0,0).second;  printf("Case %d: %d\n",cs,ans);  }  return 0;  } | using namespace std;  int aRight[mx][26],bRight[mx][26];  pp dp[mx][mx];  string s1,s2;  pp func(int l,int r){  if(l==s1.size()||r==s2.size()) return {0,1};  pp &ret = dp[l][r];  if(ret.first!=-1) return ret;  ret ={0,1};  for(int i=0;i<26;i++){  int ll=aRight[l][i];  int rr=bRight[r][i];  if(ll==-1||rr==-1) continue;  pp now= func(ll+1,rr+1);  if(ret.first<now.first+1){  ret.first=now.first+1;  ret.second=now.second;  }  else if(ret.first==now.first+1){  ret.second=(ret.second+now.second)%mod;  }  }  return ret;  } |

Distinct Palindromes-

using namespace std;

string s;

int dp[mx][mx][26];

int func(int l,int r,int alpha){

int &ret = dp[l][r][alpha];

if(ret!=-1) return ret;

if(l>r) return 0;

else if(l==r){

if(s[l]=='a'+alpha) return ret=1;

else return ret=0;

}

else if(s[l]!='a'+alpha||s[r]!='a'+alpha){

return ret = (1ll\*func(l+1,r,alpha)+1ll\*func(l,r-1,alpha)-1ll\*func(l+1,r-1,alpha))%mod;

}

else{

lli ans=2ll;

for(int i=0;i<26;i++){

ans=(ans+1ll\*func(l+1,r-1,i))%mod;

}

return ret=ans;

}

}

int main(){

// freopen("Input.txt","r",stdin); freopen("Output.txt","w",stdout);

cin>>s;

mem(dp,-1);

lli ans=0ll;

for(int i=0;i<26;i++){

ans=(ans+func(0,s.length()-1,i))%mod;

}

cout<<ans<<'\n';

return 0;

}

**/\* Dynamic Programming implementation of Edit Distance problem \*/**

|  |  |
| --- | --- |
| int main(){  // freopen("Input.txt","r",stdin); freopen("Output.txt","w",stdout);  // ios\_base::sync\_with\_stdio(false); cin.tie(NULL);  while(cin>>str1){  if(str1[0]=='#') break;  cin>>str2;  mem(dp,0);  for(int i=0;i<=str1.length();i++){  for(int j=0;j<=str2.length();j++){  if(i==0){  dp[i][j]=j;  }  else if(j==0){  dp[i][j]=i;  }  else if(str1[i-1]==str2[j-1]){  dp[i][j]=dp[i-1][j-1];  }  else{  dp[i][j]=1+min(dp[i-1][j-1],min(dp[i][j-1],dp[i-1][j]));  }  }  }  print\_path();  }  return 0;  }  /\*\*  Sample Input  abcde bcgfe  #  Sample Output  Da01Cg03If04E  \*\*/ | int dp[101][101];  string str1,str2;  void print\_path(){  int i=str1.length();  int j=str2.length();  while(i>0||j>0){  if(str1[i-1]==str2[j-1]){  i--;  j--;  }  else if(j>0&&dp[i][j]==dp[i][j-1]+1){  cout<<"I"<<str2[j-1];  if(i<=8) cout<<"0";  cout<<i+1;  j--;  }  else if(i>0&&j>0&&dp[i][j]==dp[i-1][j-1]+1){  cout<<"C"<<str2[j-1];  if(i<=9) cout<<"0";  cout<<i;  i--;  j--;  }  else if(i>0&&dp[i][j]==dp[i-1][j]+1){  cout<<"D"<<str1[i-1];  if(i<=9) cout<<"0";  cout<<i;  i--;  }  }  cout<<"E\n";  } |

MCM

int main(){

int i,j,k,d,n,p[1001],dp[101][101];

cin>>n;

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

cin>>p[i];

}

//initialization

for(i=1;i<=n;i++) dp[i][i]=0;

for(d=1;d<n;d++){

for(i=1;i<=n-d;i++){

j=i+d;

k=i;

dp[i][j]=dp[i][k]+dp[k+1][j]+p[i-1]\*p[k]\*p[j];

for(k=i+1;k<j;k++){

dp[i][j]=min(dp[i][j],dp[i][k]+dp[k+1][j]+p[i-1]\*p[k]\*p[j]);

}

// cout<<i<<" "<<j<<endl;

}

}

cout<<"Minimum Operation: "<<dp[1][n];

return 0;

}