數字三角形：

int solve(int i, int j){

if(d[i][j]>=0) return d[i][j];

return d[i][j] = a[i][j] + (i==n?0:max(solve(i+1,j),solve(i+1,j+1)));

}

矩形嵌套：

int dp(int i) { int& ans = d[i];

if(ans > 0) return ans; ans = 1;

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

if(G[i][j]) ans = max(ans, dp(j)+1);

return ans; }

多段圖的最短路之單向TSP：

int ans = INF, first = 0;

for(int j = n-1; j >= 0; j--) {

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

if(j == n-1) d[i][j] = a[i][j];

else { int rows[3] = {i, i-1, i+1};

if(i == 0) rows[1] = m-1; if(i == m-1) rows[2] = 0;

sort(rows, rows+3); d[i][j] = INF;

for(int k = 0; k < 3; k++) {

int v = d[rows[k]][j+1] + a[i][j];

if(v < d[i][j]) { d[i][j] = v; next[i][j] = rows[k]; }}}

if(j == 0 && d[i][j] < ans) { ans = d[i][j]; first = i; }}}

printf("%d", first+1);

for(int i = next[first][0], j = 1; j < n; i = next[i][j], j++) printf(" %d", i+1);

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

矩形嵌套之巴比倫塔：

int dp(int i, int j) {

int& ans = d[i][j];

if(ans > 0) return ans;

ans = 0;

int v[2], v2[2];

get\_dimensions(v, i, j);

REP(a,n) REP(b,3) {

get\_dimensions(v2, a, b);

if(v2[0] < v[0] && v2[1] < v[1])

ans = max(ans, dp(a,b));

}

ans += blocks[i][j];

return ans;

}

硬幣問題：

minv[0] = maxv[0] = 0;

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

minv[i]=INF; maxv[i]=-INF;

}

for(int i=1;i<=S;i++)

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

if(i>=V[j]){

minv[i] = min(minv[i],minv[i-V[j]]+1);

maxv[i] = max(maxv[i],maxv[i-V[j]]+1);

}

硬幣問題：

int dp(int S) { if(vis[S]) return d[S];

vis[S] = 1; int& ans = d[S]; ans = -(1<<30);

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

if(S>=V[i]) ans = max(ans, dp(S-V[i])+1);

return ans;}

0-1背包問題：

for(int i=n; i>=1; i--)

for(int j=0; j<=C; j++){

d[i][j] = (i==n?0:d[i+1][j]);

if(j>=V[i]) d[i][j]=max(d[i][j],d[i+1][j-V[i]]+W[i]);

}

**電子人的基因**

for(int i=1;i<=len1;i++)

for(int j=1;j<=len2;j++){

if(str1[i]==str2[j]){

dp1[i][j]=dp1[i-1][j-1]+1;

dp2[i][j]+=dp2[i-1][j-1];}

else{

dp1[i][j]=max(dp1[i-1][j],dp1[i][j-1]);

if(dp1[i][j]==dp1[i-1][j]) dp2[i][j]+=dp2[i-1][j];

if(dp1[i][j]==dp1[i][j-1]) dp2[i][j]+=dp2[i][j-1];}}

最短長度len1+len2-dp1[l1][l2]，有dp2[i][j]種

**最長的滑雪路徑**

int dfs(int i, int j){

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

dp[i][j] = 0;

for(int k=0; k<4; k++) { int xx = i + x[k];

int yy = j + y[k];

if(a[xx][yy]<a[i][j])

dp[i][j] = max(dp[i][j] , dfs(xx,yy)); }

return ++dp[i][j];}

**編輯距離 打印路徑**

void printA(int i,int j){

if(i\*j==0){

while(i>0){ printf("%d Delete %d\n",s++,i); --i; }

while(j>0){ printf("%d Insert %d,%c\n",s++,i+1,b[j]); --j; }

}else if(a[i]==b[j]){printA(i-1,j-1);

}else{

if(dp[i][j]-1 == dp[i-1][j-1]){

printf("%d Replace %d,%c\n",s++,i,b[j]); printA(i-1,j-1);

}else if(dp[i][j]-1 == dp[i][j-1]){

printf("%d Insert %d,%c\n",s++,i+1,b[j]); printA(i,j-1);

}else if(dp[i][j]-1 == dp[i-1][j]){

printf("%d Delete %d\n",s++,i); printA(i-1,j); }}}

**樹的最大獨立集**

#include<vector><map> cons tint maxn=205;

int cnt; int n, d[maxn][2], f[maxn][2];

vector<int> sons[maxn]; map<string, int> dict;

int ID(const string& s) {

if(!dict.count(s)) dict[s] = cnt++;return dict[s];}

int dp(int u, int k) { f[u][k] = 1; d[u][k] = k;

for(i:0~sons[u].size()-1) { int v = sons[u][i];

if(k == 1) { d[u][1] += dp(v, 0); if(!f[v][0]) f[u][1] = 0; }

else { d[u][0] += max(dp(v, 0), dp(v, 1));

if(d[v][0] == d[v][1]) f[u][k] = 0;

else if(d[v][0] > d[v][1] && !f[v][0]) f[u][k] = 0;

else if(d[v][1] > d[v][0] && !f[v][1]) f[u][k] = 0;}} return d[u][k];}

int main() { string s, s2; while(cin >> n >> s) {

cnt = 0; dict.clear(); for(int i = 0; i < n; i++) sons[i].clear();

ID(s); for(i:0~n-1) {cin>>s>>s2; sons[ID(s2)].push\_back(ID(s));}

printf("%d ", max(dp(0, 0), dp(0, 1))); bool unique = false;

if(d[0][0] > d[0][1] && f[0][0]) unique = true;

if(d[0][1] > d[0][0] && f[0][1]) unique = true;

if(unique) printf("Yes\n"); else printf("No\n");}}

**編輯距離**

int solve(int i,int j){ if(dp[i][j]==-1){ if(i\*j==0) dp[i][j] = i+j;

else if(a[i]==b[j]) dp[i][j] = findDp(i-1,j-1);

else dp[i][j] = min(solve(i-1,j-1),min(solve(i-1,j),solve(i,j-1)))+1;}

return dp[i][j];}

**最大連續和** A在[x,y)中的最大連續和

int maxsum(int\* A,int x,int y){

int v,L,R,maxs;

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

int m=x+(y-x)/2;

int maxs =max(maxsum(A,x,m),maxsum(A,m,y);

int v,L,R; v=0; L=A[m-1];

for(int i=m-1;i>=x;i--) L=max(L,v+=A[i]);

return max(maxs,L+R);}

**LCS dp[1~l1][1~l2]=0**

for(i:1~l1) for(j:1~l2)

{ if(s1[i]==s2[j]) dp[i][j]=dp[i-1][j-1]+1;

else {dp[i][j]=max(dp[i][j-1],dp[i-1][j]);} }

**切木棍 a1~an切的位置 dp(0,n-1)**

int dp(int i, int j) {

if(i >= j - 1) return 0;if(vis[i][j]) return d[i][j];

vis[i][j] = 1; int& ans = d[i][j]; ans = -1;

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

int v = dp(i,k) + dp(k,j) + a[j] - a[i];

if(ans<0||v<ans) ans = v;}return ans;}