**KOSARAJU**

int g[] , g\_rev[]

dfs1(){

………..

push\_order

}

dfs2(){

comp\_push

……...

}

for(all dfs(1))

reset(vis)

reverse(order)

for(all x:order){

if(!vis[x])

dfs2(x);

print(comp);clear(comp);

}

**height and depth**

dfs(){

…

…….for(){

……... depth

……….dfs()

……….max(cur,height[x]+1);

……}

….

}

**LCA**

store par

last\_common\_par

**KNAPSACK\_DP**

vector<int>profit = {30, 50, 60};

vector<int>weight = {3, 4, 5};

int W = 8;

int dp[W + 1];

memset(dp, 0, sizeof dp);

for (auto i = 1; i < int(profit.size()) + 1; ++i)

for (auto w = W; w >= 0; --w)

if (weight[i - 1] <= w)

dp[w] = max(dp[w], profit[i - 1] + dp[w - weight[i - 1]]);

cout << dp[W] << '\n';

**LCS**

string c\_lcs(const string &s1, const string &s2) {

int n = s1.size();

int m = s2.size();

int dp[n + 1][m + 1];

memset(dp, 0, sizeof(dp));

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

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

if (s1[i - 1] == s2[j - 1])

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

else

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

string con;

while (n > 0 && m > 0) {

if (s1[n - 1] == s2[m - 1]) {

con.push\_back(s1[n - 1]);

n--, m--;

} else if (dp[n - 1][m] > dp[n][m - 1]) {

n--;

} else {

m--;

}

}

return string(con.rbegin(), con.rend());

}

**DP\_COINCHANGE**

int ammount = 11;

vector<int>coins = {1, 5, 7};

vector<int>dp(ammount + 1, ammount + 1);

dp[0] = 0;

for (auto a = 1LL; a <= ammount; ++a)

for (auto c : coins)

if (a - c >= 0)

dp[a] = min(dp[a], dp[a - c] + 1);

cout << dp[ammount] << '\n';

**KNAPSACK\_FRAC**

pair<weith,weight/val>

sort(return a.second>b.second)

for(all i){

if(ff+cur < cap){

tot+=ss;

cur+=ff;

}else{

int frac = cap – cur;

tot+=fact\*ss;

break;

}

}

**EDIT\_D**

const int INF = int(1e9) + 5;

int edit\_distance(const string &S, const string &T) {

int n = int(S.size());

int m = int(T.size());

vector<int> dp(m + 1, INF);

iota(dp.begin(), dp.end(), 0);

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

int diag = dp[0];

dp[0] = i + 1;

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

int temp = dp[j + 1];

dp[j + 1] = min({dp[j] + 1, dp[j + 1] + 1, diag + (S[i] != T[j])});

diag = temp;

}

}

return dp[m];

}

**MST**

dsu

sort(edges)assending

for(all[u,v]:edges)){

if(par[u] == par[v])cont;

now we can add this edges to the mst

}

**DIJKS**

dis[]

vis[]

par[]

MIN\_HEAP[]

……..vis[](dont vis now)

while(empt){

auto[w,v]=f; pop;

vis[]?cont:vis[]=true;

for(auto [c\_v,c\_w]:g[v]){

if(dis[v]+c\_w<dis[c\_v]) ---> update new distance

}

}