CSE246

Section 4

Fall 2023

**Lab Task - 04**

**Topic:** Complex Coin Change

Complex Coin Change with Coin Print

0/1 Knapsack

LCS and Path Print

LIS, Longest Increasing Subsequence

Hill Climbing

Submitted By

|  |  |
| --- | --- |
| **Name** | **ID** |
| B M Shahria Alam | 2021-3-60-016 |
| Golam Kibria | 2021-3-60-215 |
| Sidratul Moontaha | 2021-3-60-048 |
| MD Imran Khan | 2021-3-60-206 |

**Problem 1**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main() {

int n, k;

cin >> n >> k;

int coins[n];

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

cin >> coins[i];

}

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

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

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

if(j == 0) {

dp[i][j] = 0;

} else {

dp[i][j] = INT\_MAX-1;

}

}

}

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

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

if(coins[i-1] > j) {

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

} else {

dp[i][j] = min(dp[i-1][j], dp[i][j-coins[i-1]]+1);

}

}

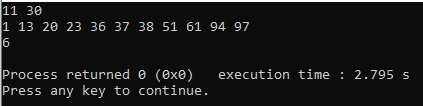
}

cout << dp[n][k] << endl;

return 0;

}

**OUTPUT:**

****

**Problem 2**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int coinCount[10007];

int coins[31];

void COIN(int n, int k)

{

int temp1[n+1],temp2[n+1];

temp1[0]=0;

temp2[0]=0;

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

{

int min=INT\_MAX;

int coin=0;

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

{

if (j>=coins[i])

{

if ((1+temp1[j-coins[i]])<min)

{

min=1+temp1[j-coins[i]];

coin=i;

}

}

}

temp1[j]=min;

temp2[j]=coin;

}

int l=n;

while (l>0)

{

coinCount[coins[temp2[l]]]++;

l=l-coins[temp2[l]];

}

for (int i=1; i<=10000; i++)

{

if (coinCount[i]>0)

{

cout<<i<<" "<<coinCount[i]<<endl;

}

}

}

int main()

{

int n,k;

cin>>n>>k;

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

{

cin>>coins[i];

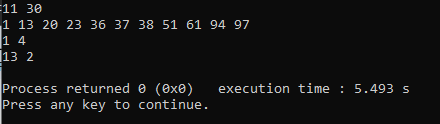
}

COIN(k,n);

return 0;

}

**OUTPUT**

****

**Problem 3**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int weight[31];

int value[31];

int knapsack(int i, int w)

{

if(i==0)

{

if(weight[0]<=w) return value[0];

return 0;

}

int r1=knapsack(i-1,w);

int r2=INT\_MIN;

if(weight[i]<=w)

{

r2=value[i]+knapsack(i-1, w-weight[i]);

}

return max(r1, r2);

}

int main()

{

int n,w;

cin>>n>>w;

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

{

cin>>value[i];

}

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

{

cin>>weight[i];

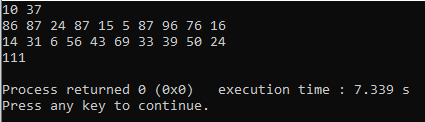
}

cout<<knapsack(n-1, w)<<endl;

return 0;

}

**OUTPUT:**



**Problem 4**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main() {

string M, N;

cin >> M >> N;

int M\_len = M.length();

int N\_len = N.length();

vector<vector<int>> dp(M\_len+1, vector<int>(N\_len+1, 0));

for(int i=1; i<=M\_len; i++) {

for(int j=1; j<=N\_len; j++) {

if(M[i-1] == N[j-1]) {

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

} else {

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

}

}

}

int lcs\_len = dp[M\_len][N\_len];

cout << lcs\_len << endl;

string lcs = "";

int i = M\_len, j = N\_len;

while(i > 0 && j > 0) {

if(M[i-1] == N[j-1]) {

lcs = M[i-1] + lcs;

i--;

j--;

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

i--;

} else {

j--;

}

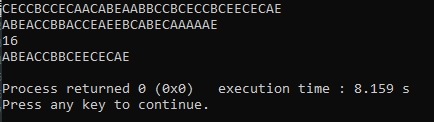
}

cout << lcs << endl;

return 0;

}

**OUTPUT:**



**Problem 5**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main() {

int N;

cin >> N;

vector<int> arr(N);

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

cin >> arr[i];

}

vector<int> dp(N, 1);

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

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

if(arr[i] > arr[j]) {

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

}

}

}

int result = 1;

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

result = max(result, dp[i]);

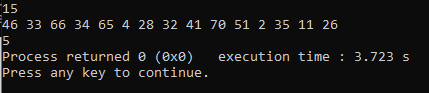
}

cout<<result;

return 0;

}

**OUTPUT:**



**Problem 6**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

int main() {

int M, N;

cin >> M >> N;

vector<vector<int>> grid(M, vector<int>(N));

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

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

cin >> grid[i][j];

}

}

for (int i = M - 2; i >= 0; --i) {

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

int best\_move = grid[i + 1][j];

if (j > 0) {

best\_move = min(best\_move, grid[i + 1][j - 1]);

}

if (j < N - 1) {

best\_move = min(best\_move, grid[i + 1][j + 1]);

}

grid[i][j] += best\_move;

}

}

int min\_danger = grid[0][0];

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

min\_danger = min(min\_danger, grid[0][j]);

}

cout << min\_danger << endl;

return 0;

}

**OUTPUT:**

