Міністерство освіти і науки України

Національний Технічний Університет

«Харківський Політехнічний Інститут»

Кафедра «Стратегічного Керування»

Лабораторна робота № 8

«ДИНАМІЧНЕ ПРОГРАМУВАННЯ»

Перевірила: Мошко Є. О.

Виконав:

Харків, 2018

Лабораторна робота № 8

Тема: Динамічне Програмування

Завдання

Розробити програму, яка читає з клавіатури вхідні дані та розв’язує задачу методом динамічного програмування. Визначити складність алгоритму.

Варіанти завдань

(2 варіант)

Пошук найбільшої спільної підпослідовності. Вхідні дані: натуральні числа N, M (1< N, M < 256) та дві послідовності X та Y натуральних чисел довжиною N та M відповідно. Вихідні дані: динамічна таблиця (𝐴(𝑖,𝑗) = довжина НСП для префіксів 𝑋𝑖та 𝑌𝑖) та НСП для X та Y.

Хід роботи

|  |
| --- |
| #include<iostream>  #include<cstring>  #include<cstdlib>  using namespace std;    /\* Returns length of LCS for X[0..m-1], Y[0..n-1] \*/  int\* lcs(int\* arr1, int\* arr2, 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 (arr1[i-1] == arr2[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  int\* lcs = new int[index+1];  lcs[0]=index;  // 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 (arr1[i-1] == arr2[j-1])  {  lcs[index] = arr1[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--;  }  return lcs;  }  int main() {  int M, N;  int value;  int size;  N=4;  M=8;  int\* arr1 = new int[M]{1,2,3,4,7,7,7,8};  int\* arr2 = new int[N]{1,2,4,8};  // for (int i=0; i<M; i++){  // cin >> arr1[i];  // }  // for (int i=0; i<N; i++){  // cin >> arr1[i];  // }  int\* arr = lcs(arr1,arr2,M,N);  cout << "Length of LCS: " << arr[0] << endl;  for (int i=1; i<=arr[0]; i++){  cout << arr[i];  }  } |

Приклад виконання програми:

|  |
| --- |
| Length of LCS: 4  1248  --------------------------------  Process exited after 0.07627 seconds with return value 0  Press any key to continue . . . |

Складність алгоритму дорівнює O(M⋅N) де, M та N – довжини послідовностей.

Висновок

Динамічне програмування - метод оптимізації, пристосований до операцій, в яких процес прийняття рішень може бути розділений на окремі етапи (кроки). Одними з класичних прикладів його використання є пошук найбільшої спільної послідовності (ця лаба), пошук траєкторії, розвязвання задачі про добуток матриць.