МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп’ютерних наук та штучного інтелекту

ПРАКТИЧНА РОБОТА №3

на тему: «Динамічне програмування»

Виконав: студент 2 курсу групи КС22
Спеціальності 122 «Комп’ютерні науки»
Скрипняк Тарас Артемович
Прийняв: викладач
Олешко О.І.

Харків – 2024

Завдання 1. На прямій дощечці вбиті гвіздки. Будь-які два гвіздки можна з'єднати ниточкою. Потрібно з'єднати деякі пари гвіздків ниточками так, щоб до кожного гвіздка була прив'язана хоча б одна ниточка, а сумарна довжина всіх ниточок була мінімальна.

Вхідні дані: У першому рядку вхідного потоку записане число N - кількість гвіздків (2 <= N <= 100). У наступному рядку записано N чисел - координати всіх гвіздків (невід’ємні цілі числа, що не перевершують 10000).

Вихідні дані: мінімальна сумарна довжина всіх ниточок.

```
#include <stdio.h>

int main() {
    int N;

    printf("Enter amount of nails: ");
    scanf("%d", &N);
    if (N < 2 || N > 100) return printf("Nails can't exceed 100 or subseed 2\n");

    // масив з координатами гвіздків
    int a[N];

    printf("Enter nail coordinates (separated by a space):\n");
    for (int i = 0; i < N; i++) {
        scanf("%d", &a[i]);
        if (a[i] < 0 || a[i] > 10000) return printf("Nail coordinates must be >0 and <=10000\n");
    }

    // це алгоритм сортування гвіздків за зростанням координат
    // сортуємо ми через те, що порядок гвіздків не потрібен для обчислення суми відстаней
    // а за зростанням обчислювати відстані набагато легше
    for (int i = 0; i < N - 1; i++)
        for (int j = 0; j < N - i - 1; j++)
            if (a[j] > a[j + 1]) {
                int temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
            // цей else перевіряє на наявність гвіздків з однаковими координатами
            else if (a[j] == a[j+1])
                return printf("Nails with same coordinates are not allowed\n");

    /* це масив з відстанями для кожного гвіздка
       структура масиву: кожен індекс масиву відповідає кожному гвіздку в масиві з координатами
       кожен елемент масиву - це відстань від відповідного гвіздка від попереднього

       це зроблено тому, що ця структура - найлегший варіант реалізації для обчислення мінімальних
дистанцій
       бо дивлячись на елементи масиву можна легко візуалізувати як конкретно з'єднані ниточки між
гвіздками

       візьмемо 6 гвіздків: 3, 4, 6, 12, 13, 14. логічно, що з'єднані 3-4, 4-6, 12-13, 13-14, та відстань
буде 5.
       тоді як виглядають ці два масива:
       a[6] = {3, 4, 6, 12, 13, 14};
       d[6] = {0, 1, 2, 0, 1, 1};


       тобто відстань між другим гвіздком та першим - 1, між третім та другим - 2 і т.д.
       це зроблено тому, що
       1. це дуже просто розуміти та зберігати. беручі елементи масиву d та координату лише одного з
гвіздків
            ми за допомогою цього масиву можем реконструювати масив a (координати інших гвіздків).
       2. це дозволяє дуже просто обчислити сумарну мінімальну відстань, просто додавши всі елементи масиву

       суть виконанного це все не міняє - програма так само буде правильно вичисляти мінімальну сумарну
відстань
       масиву d можна в цілому і позбавитися, але т.я. на практичному занятті він був присутній, я його
імплементував
    */
    int d[N];
```

```
    // через те, що перший (по порядку зростання) гвіздок
    // НІКОЛИ НЕ МАЄ ПОПЕРЕДНЬОГО, відстань від нього до попереднього завжди 0
    d[0] = 0;
    // передостанній гвоздик нуль зараньше для правильних обчислення суми, а також тому, що він автоматично з'єднаний з останнім
    d[N-2] = 0;
    // тут ми обчислюємо відстань від другого гвізда до першого, та від останього до попереднього.
    // зроблено це через те, що другий гвіздок і останній ЗАВЖДИ з'єднані з попередніми, тобто першим та передостаннім
    // тоді нам не треба обчислювати їх у подальшому циклі і просто зробити це вручну
    d[1] = a[1] - a[0];
    d[N-1] = a[N-1] - a[N-2];


    // цей цикл вичисляє відстані від третього гвіздка до перед-передостаннього
    for (int i = 2; i < N - 2; i++) {
        // тут ми обчислюємо відстань від вибранного циклом гвіздка до попереднього
        // зроблено це для того, щоб переконатися в тому, що гвіздок і має ХОЧА-Б ОДНУ НИТОЧКУ
        int x = a[i+1]  - a[i],
            y = a[i]    - a[i-1];


        // якщо відстань від гвіздка до наступного менша за відстань до попереднього
        // вводимо значення відстані до наступного у елемент масиву, відповідному наступному гвіздку
        // (див. структура масиву)
        if (x < y) { d[i] = 0; d[i + 1] = x; }
        // інакше, записуємо в масив відстаней на індекс теперішнього гвіздка відстань між ним і попереднім
        else if (x > y) d[i] = y;

        // це "або" перевіряють варіант, в якому відстані однакові. тоді програма вибере той гвіздок, що ще не з'єднаний ниткою
        // це превентує кейси коли координати щось наприклад
        // { 3, 4, 5, 6, 7... }
        // і замість з'єднання всіх підряд, тобто
        // { 1, 1, 1, 1, 1... }
        // програма зробить
        // { 0, 1, 0, 1, 0... }
        else {
            if (d[i-1]) { d[i] = 0; d[i + 1] = x; }
            else d[i] = y;
        }


    }

    // виводимо суму мінімальних відстаней ниточок
    int sum = 0;
    for(int i = 0; i < N; i++)
        sum += d[i];

    printf("Minimum sum of string length: %d\n", sum);

    return 0;
}
```

Лістинг

```
bouncytorch@AORUS:~/Repos/homework-c/pr3/tarik$ ./a.out
Enter amount of nails: -3
Nails can't exceed 100 or subseed 2
bouncytorch@AORUS:~/Repos/homework-c/pr3/tarik$ ./a.out
Enter amount of nails: 2
Enter nail coordinates (separated by a space):
-1 -2
Nail coordinates must be >0 and <=10000
bouncytorch@AORUS:~/Repos/homework-c/pr3/tarik$ ./a.out
Enter amount of nails: 6
Enter nail coordinates (separated by a space):
3 4 12 6 13 14
Minimum sum of string length: 5
```

Результати виконання