ДОДАТОК А

Графічний матеріал атестаційної роботи

Харківський національний університет радіоелектроніки

Кафедра ЕОМ

Модель розумної зупинки для Smart Cities

Атестаційна робота

Другий (магістерський) рівень

Автор:
Славтіч Д.О.
студ. гр. СПм-19-1

Керівник:
Токарєв В.В.
доц. каф. ЕОМ

МЕТА І ЗАДАЧІ РОБОТИ

2

МЕТОЮ АТЕСТАЦІЙНОЇ РОБОТИ є розробка алгоритму пошуку короткого шляху від точки А до точки В.

ЗАВДАННЯ ДЛЯ ДОСЯГНЕННЯ ПОСТАВЛЕНОЇ МЕТИ:

- провести огляд моделей Smart Cities;
- провести огляд моделей інтелектуальних транспортних систем - ITC;
- провести огляд «розумних зупинок».

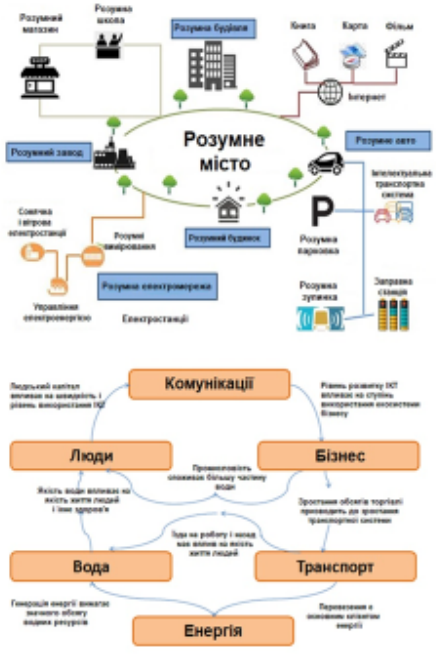
АКТУАЛЬНІСТЬ РОБОТИ

3

В даний час інформаційні технології застосовуються у все більшій кількості сфер життя людини. Однією з актуальних областей наукових досліджень є сфера життєвого оточення, яка з області Smart House розвивається в даний час в область Smart city, Smart transport system. Було виявлено, що на поточному етапі не існує універсальної моделі Smart city і точного її визначення. Одиією з основних проблем в контексті всього міста, було виявлено цілий ряд проблем у транспортній сфері. Один із шляхів рішень - розробка інтелектуальної транспортної системи. Системи «розумної зупинки» є компонентою ITC. Таким чином, стає очевидним, що термінал «розумної зупинки» може бути мікромоделлю еволюційного переходу міста до стану Smart city.

ЗАГАЛЬНА ІНФРАСТРУКТУРА І КОМПОНЕНТИ SMART CITY

4



В даний час ведуться розробки по застосуванню нової моделі розвитку міст - реалізація концепції Smart city, яка в своїй основі застосовує ІКТ для вирішення всіх сфер життєдіяльності населення. Така модель має три основні версії, але ідея управління містами з максимально ефективним використанням природних ресурсів та забезпеченням високого рівня життя єдина для всіх. Міста стають не тільки місцем проживання, а й «розумною» територією, комфортним і дружнім середовищем для людини.

КЛЮЧОВІ ХАРАКТЕРИСТИКИ ДЛЯ SMART CITY ТАКІ: 5

Smart city зобов'язане запропонувати гідний рівень життя кожному жителю мегаполісу.

Це означає надати високу якість життя:

- чисте повітря;
- якісну освіту;
- безпеку;
-недорогу високоякісну охорону здоров'я;
- розваги;
- спорт;
- ефективну міську мобільність, і високу швидкість взаємодії.

Ключові характеристики для Smart city такі:

-конкурентоспроможність;

-стійкість;

-якість життя.

МОДЕЛІ SMART CITY 6

Архітектура моделі Smart city згідно IBM

Архітектура моделі Smart city згідно Microsoft

Архітектура моделі Smart city згідно Accenture

СЕМАНТИЧНА МЕРЕЖА SMART CITY

7

Як показав проведений аналіз ініціатив, було визначено, що дані ініціативи можуть мати відповідність до положень процедурам побудови Smart city згідно Європейської моделі. Одна з ініціатив - розробка ІТС, а саме інтелектуальних зупинок.

СЕМАНТИЧНА МЕРЕЖА РОЗУМНОЇ ЗУПИНКИ

8

Для визначення модулів "Розумної зупинки" був проведений аналіз можливих варіантів впровадження апаратних і програмних рішень, які при інтеграції можуть створити платформу для переходу до Smart City.

ЗАГАЛЬНА КОНЦЕПЦІЯ «РОЗУМНОЇ ЗУПИНКИ»

9

«Розумна зупинка» в Барселоні

МОДЕЛЬ «РОЗУМНОЇ ЗУПИНКИ»

10

ЗАГАЛЬНА КОНЦЕПЦІЯ «РОЗУМНОЇ ЗУПИНКИ»

11

Розумний термінал

Навігаційна структура системи «розумної зупинки»

ПРИКЛАД РОЗМІЩЕННЯ ЗУПИНОЧНИХ ПУНКТІВ ОДНОГО З АВТОБУСНИХ МАРШРУТІВ

12

РОЗРОБЛЕНИЙ АЛГОРИТМ ПОШУКУ МАРШРУТІВ ВІД ТОЧКИ А ДО ТОЧКИ В

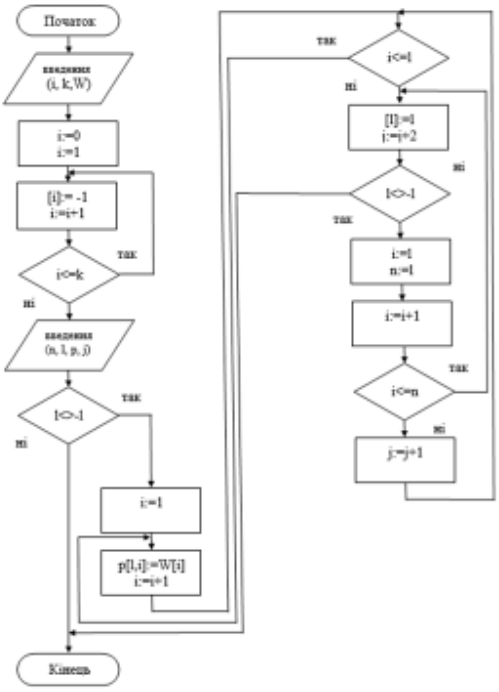
13

Крок №1. Знайдено маршрути без пересадок.

Крок №2. Знайдений один маршрут з однією пересадкою.

Крок №3. Знайдений один маршрут з двома пересадками.

Крок №4. Знайдений один маршрут з трьома або більше пересадками.

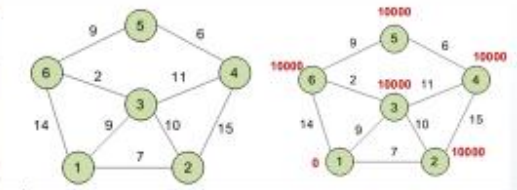


БЛОК-СХЕМА АЛГОРИТМА ПОШУКУ МАРШРУТІВ ВІД ТОЧКИ А ДО ТОЧКИ В

14

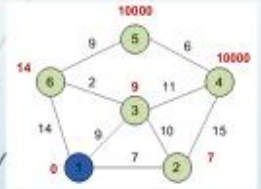
АЛГОРИТМ ДЕЙКСТРИ

15

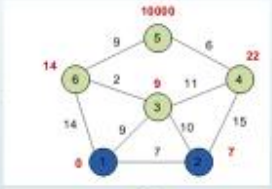


Ініціалізація алгоритму Дейкстри

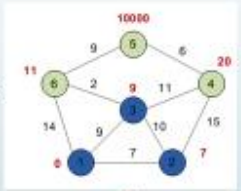
Крок №1



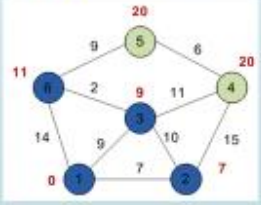
Крок №2



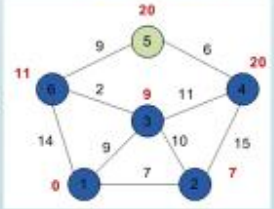
Крок №3



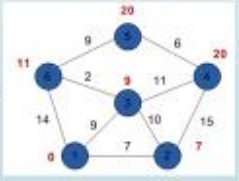
Крок №4



Крок №5



Крок №6



ВИСНОВКИ

16

ПІД ЧАС ВИКОНАННЯ МАГІСТЕРСЬКОЇ АТЕСТАЦІЙНОЇ РОБОТИ БУВ розроблений алгоритм, який здійснює пошук короткого маршруту від точки А до точки В.

В МАГІСТЕРСЬКІЙ АТЕСТАЦІЙНІЙ РОБОТІ ВИРІШЕНІ ТАКІ ЗАДАЧІ:

- проведено огляд моделей Smart Cities;
- проведено огляд моделей інтелектуальних транспортних систем - ITC;
- огляд «розумних зупинок».

17

ДЯКУЮ ЗА УВАГУ

ДОДАТОК Б

Вихідний код програми

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#define SIZE 6
int main()
{
  int a[SIZE][SIZE]; // матриця зв'язків
  int d[SIZE]; // мінімальна відстань
  int v[SIZE]; // відвідані вершини
  int temp, minindex, min;
  int begin_index = 0;
  system("chcp 1251");
  system("cls");
  // Ініціалізація матриці зв'язків
  for (int i = 0; i<SIZE; i++)
  {
    a[i][i] = 0;
    for (int j = i + 1; j<SIZE; j++) {
      printf("Введіть відстань %d - %d: ", i + 1, j + 1);
      scanf("%d", &temp);
      a[i][j] = temp;
      a[j][i] = temp;
    }
  }
  // Вивід матриці зв'язків
  for (int i = 0; i<SIZE; i++)
  {
    for (int j = 0; j<SIZE; j++)
      printf("%5d ", a[i][j]);
    printf("\n");
  }
  // Ініціалізація вершин і відстаней
  for (int i = 0; i<SIZE; i++)
  {
    d[i] = 10000;
    v[i] = 1;
  }
  d[begin_index] = 0;
  // Крок алгоритму
  do {
    minindex = 10000;
    min = 10000;
    for (int i = 0; i<SIZE; i++)
    { // Якщо вершину ще не оминули й вага менше min
      if ((v[i] == 1) && (d[i]<min))
      { // Переприсвоюємо значення
        min = d[i];
        minindex = i;
      }
    }
    // Додаємо знайдену мінімальну вага
    // до поточної ваги вершини
    // і порівнюємо з поточною мінімальною вагою вершини
    if (minindex != 10000)
    {
      for (int i = 0; i<SIZE; i++)
      {
```

```
            if (a[minindex][i] > 0)
            {
              temp = min + a[minindex][i];
              if (temp < d[i])
              {
                d[i] = temp;
              }
            }
          }
          v[minindex] = 0;
        }
      } while (minindex < 10000);
      // Вивід найкоротших відстаней до вершин
      printf("\nНайкоротша відстань до вершин: \n");
      for (int i = 0; i<SIZE; i++)
        printf("%5d ", d[i]);

      // Відновлення шляху
      int ver[SIZE]; // масив відвіданих вершин
      int end = 4; // індекс кінцевої вершини = 5 - 1
      ver[0] = end + 1; // початковий елемент - кінцева вершина
      int k = 1; // індекс попередньої вершини
      int weight = d[end]; // вага кінцевої вершини

      while (end != begin_index) // поки не дійшли до початкової вершини
      {
        for (int i = 0; i<SIZE; i++) // переглядаємо всі вершини
          if (a[i][end] != 0)   // якщо зв'язок є
          {
            int temp = weight - a[i][end]; // визначаємо вагу шляху з
попередньої вершини
            if (temp == d[i]) // якщо вага збіглася з розрахованою
            {                 // значить з цієї вершини і був перехід
              weight = temp; // зберігаємо нову вагу
              end = i;       // зберігаємо попередню вершину
              ver[k] = i + 1; // і записуємо її в масив
              k++;
            }
          }
      }
      // Вивід шляху (початкова вершина виявилася в кінці масиву з k
елементів)
      printf("\nВивід найкоротшого шляху \n");
      for (int i = k - 1; i >= 0; i--)
        printf("%3d ", ver[i]);
      getchar(); getchar();
      return 0;
    }
```