ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện CNTT & Truyền thông



**BÁO CÁO PROJECT 2 C ADVANCED**

**Nhóm 5:**

1. Trần Đức Trung – 20176893
2. Nguyễn Văn Giang – 20176740
3. Đỗ Thành Nam – 20176829
4. Nguyễn Tuấn Việt – 20176909

**MỤC LỤC**

[Lời mở đầu 3](#_Toc8801321)

[PHẦN I: YÊU CẦU BÀI TẬP LỚN 4](#_Toc8801322)

[PHẦN II: ĐỊNH HƯỚNG GIẢI QUYẾT BÀI TOÁN - PHÂN CHIA CÔNG VIỆC 4](#_Toc8801323)

[PHẦN III: CÁC HÀM CHÍNH SỬ DỤNG 5](#_Toc8801324)

[1. Hàm xử lý File 5](#_Toc8801325)

[2. Hàm tìm đường đi ngắn nhất (Dijkstra) 7](#_Toc8801326)

[PHẦN IV: DEMO CHƯƠNG TRÌNH 10](#_Toc8801327)

[1. Lưu trữ dữ liệu bản đồ 10](#_Toc8801328)

[2. Tìm kiếm đường đi ngắn nhất 11](#_Toc8801329)

[3. Demo tìm đường đi với các tuyến Bus thực tế 12](#_Toc8801330)

# Lời mở đầu

Bản báo cáo khái quát quá trình thực hiện bài tập lớn, gồm có 5 phần:

* Phần I trình bày yêu cầu của bài tập lớn.
* Phần II định hướng cách giải quyết bài tập và phân công công việc cho từng người trong nhóm.
* Phần III gồm code của các hàm chính.
* Phần IV là hình ảnh demo chương trình do nhóm thực hiện.

Chương trình do nhóm viết không tham khảo code ngoài việc sử dụng thư viện JRB và cài đặt thuật toán Dijkstra do giảng viên cung cấp, dữ liệu bus được thu thập trên trang web timbus.vn.

Chương trình và bản báo cáo chắc chắn sẽ không tránh khỏi những sai sót. Cả nhóm rất mong nhận được ý kiến của thầy giáo và các bạn.

Xin chân thành cảm ơn.

Hà Nội, ngày 15 tháng 5 năm 2019

# PHẦN I: YÊU CẦU BÀI TẬP LỚN

Viết một chương trình mô phỏng bản đồ bus tại Hà Nội

1. Tổ chức và lưu trữ dữ liệu trong 1 file nạp vào chương trình khi chạy
2. Viết lại Graph API để lưu lại bản đồ bus trong bộ nhớ
3. Xây dựng hàm để tìm đường đi ngắn nhất giữa 2 bến

# PHẦN II: ĐỊNH HƯỚNG GIẢI QUYẾT BÀI TOÁN - PHÂN CHIA CÔNG VIỆC

1. Định hướng giải quyết bài toán:

* Tìm kiếm và tạo một file dữ liệu bản đồ bus tại Hà Nội.
* Sử dụng thư viện JRB – Tree để viết các hàm tìm kiếm/thêm/sửa/xóa.

*(\*)Tại sao lại dùng B-tree:*

*- Trong quá trình làm việc, bọn em nghĩ là cần một loại cấu trúc dữ liệu có tính chất đối xứng và có thứ tự để cho truy cập tới mọi định nghĩa từ đều có thời gian giống nhau. Một trong những giải pháp là B-tree.*

*- Dùng B-tree có thể truy cập trực tiếp một lướng lớn từ trên đĩa mềm mà không tốn RAM,nhờ cấu trúc đặc trưng của B-tree nên thời gian truy cập đến các từ nhanh,đều như nhau hoặc chênh nhau rất ít.*

* Phải có hỗ trợ hoàn thành tìm kiếm.
* Sử dụng soundex library để gợi ý tìm kiếm.
* Test chương trình với các kích thước từ điển khác nhau.
* Viết báo cáo .

1. Phân chia công việc:

* Đỗ Thành Nam: Đưa ra cấu trúc lưu trữ dữ liệu và viết Menu
* Trần Đức Trung: Thu thập dữ liệu, tạo file và xử lý file
* Nguyễn Văn Giang: Làm báo cáo
* Nguyễn Tuấn Việt: Viết thuật toán tìm đường

# PHẦN III: CÁC HÀM CHÍNH SỬ DỤNG

## 1. Hàm xử lý File



void DocFile(FILE \*f, Graph g)

{

data \*p;

p = malloc(sizeof(data));

f = fopen("input1.txt", "r");

char tmp[100];

fgets(tmp, 100, f);

char ten\_ben\_1[100];

ten\_ben\_1[0] = '\0';

char ten\_tuyen[100];

char ten\_ben\_2[100];

ten\_ben\_2[0] = '\0';

while(tmp[0] != '?')

{

fscanf(f, "%s", ten\_tuyen);

fscanf(f, "%s", tmp);

fscanf(f, "%s", tmp);

fscanf(f, "%s", tmp);//doc cum dau tien cua ten ben

while(1)

{

while(tmp[0] != '-' && tmp[0] != '.' && tmp[0] != '?')

{

strcat(ten\_ben\_1, tmp);

strcat(ten\_ben\_1, " ");

fscanf(f, "%s", tmp);

}

if( jrb\_find\_str(g.vertices, strdup(ten\_ben\_1)) == NULL)

{

addVertex(g, strdup(ten\_ben\_1), so\_ben++);

}

tmp[0] = '\0';

fscanf(f, "%s", tmp);

while(tmp[0] != '-' && tmp[0] != '.' && tmp[0] != '?')

{

strcat(ten\_ben\_2, tmp);

strcat(ten\_ben\_2, " ");

fscanf(f, "%s", tmp);

}

if( jrb\_find\_str(g.vertices, strdup(ten\_ben\_2)) == NULL)

{

addVertex(g, strdup(ten\_ben\_2), so\_ben++);

}

addEdge(g, ten\_ben\_1, ten\_ben\_2, ten\_tuyen);

themCanh(g, getVertex(g, strdup(ten\_ben\_1)), getVertex(g, strdup(ten\_ben\_2)), 1);

if (tmp[0] == '.' || tmp[0] == '?')

{

ten\_tuyen[0] = '\0';

ten\_ben\_1[0] = '\0';

ten\_ben\_2[0] = '\0';

break;

}else

{

fseek(f, -((strlen(ten\_ben\_2) -1) + 2), SEEK\_CUR);//vi luc nay ten\_ben\_2 co dau ' ' o cuoi

fscanf(f, "%s", tmp);

}

ten\_ben\_1[0] = '\0';

ten\_ben\_2[0] = '\0';

}

}

}

## 2. Hàm tìm đường đi ngắn nhất (Dijkstra)

double ngan\_nhat(Graph graph, int s, int t, int \*path, int \*length)

{

// Khoi tao cac distance = 0

double distance[1000], min;

int previous[1000], u, visit[1000];

for (int i=0; i<1000; i++){

distance[i] = INFINITIVE\_VALUE;

visit[i] = 0;

previous[i] = 0;

}

distance[s] = 0;

previous[s] = s;

visit[s] = 1;

Dllist ptr, queue, node;

queue = new\_dllist();

dll\_append(queue, new\_jval\_i(s));

// Duyet Queue

while (!dll\_empty(queue)){

min = INFINITIVE\_VALUE;

dll\_traverse(ptr, queue){

// Lay ra min{distance}

u = jval\_i(ptr->val);

if (min > distance[u]){

min = distance[u];

node = ptr;

}

}

u = jval\_i(node->val);

visit[u] = 1;

dll\_delete\_node(node);

if (u == t) break;

int output[100];

int n = layDinhKe(graph, u, output);

// Cap nhap distance cua tat ca cac dinh ma lien ke voi dinh min

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

int v = output[i];

double w = layGiaTriCanh(graph, u, v);

if (distance[v] > distance[u] + w){

distance[v] = distance[u] + w;

previous[v] = u;

}

if (visit[v] == 0){

dll\_append(queue, new\_jval\_i(v));

}

}

}

// Truy vet lai de lay duong di tu s ---> t va luu trong path[]

double distance\_s\_t = distance[t];

int count=0;

if (distance[t] != INFINITIVE\_VALUE){

path[count++] = t;

while (t != s){

t = previous[t];

path[count++] = t;

}

\*length = count;

}

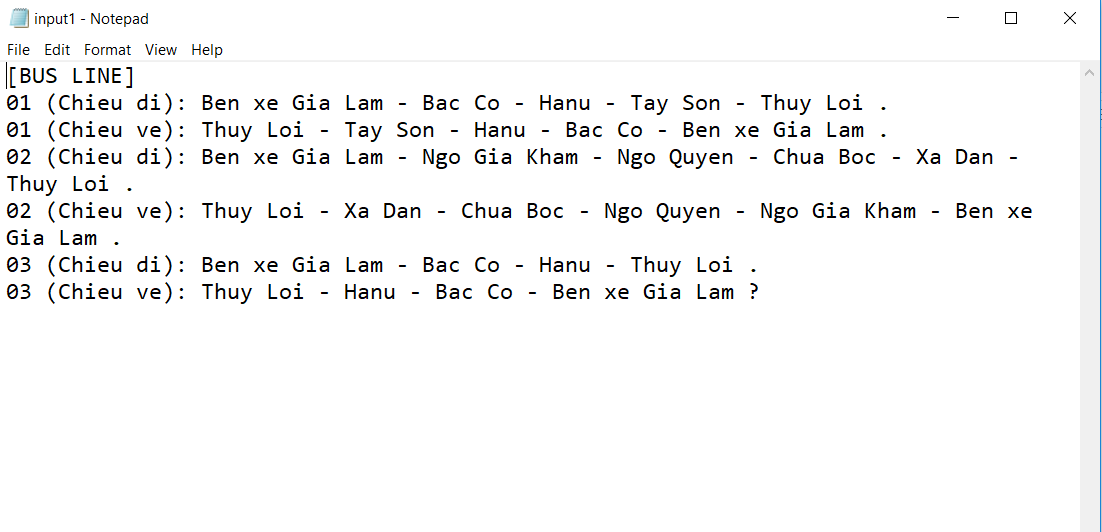
return distance\_s\_t;

}

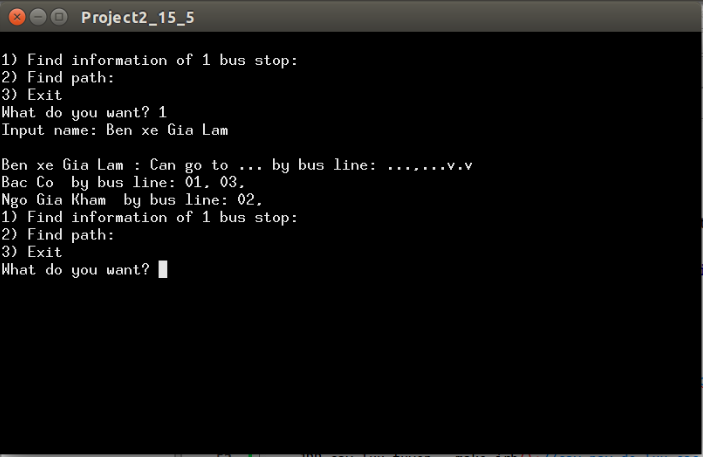
# PHẦN IV: DEMO CHƯƠNG TRÌNH

## 1. Lưu trữ dữ liệu bản đồ

\_ Kiểm thử độ chính xác với dữ liệu nhỏ

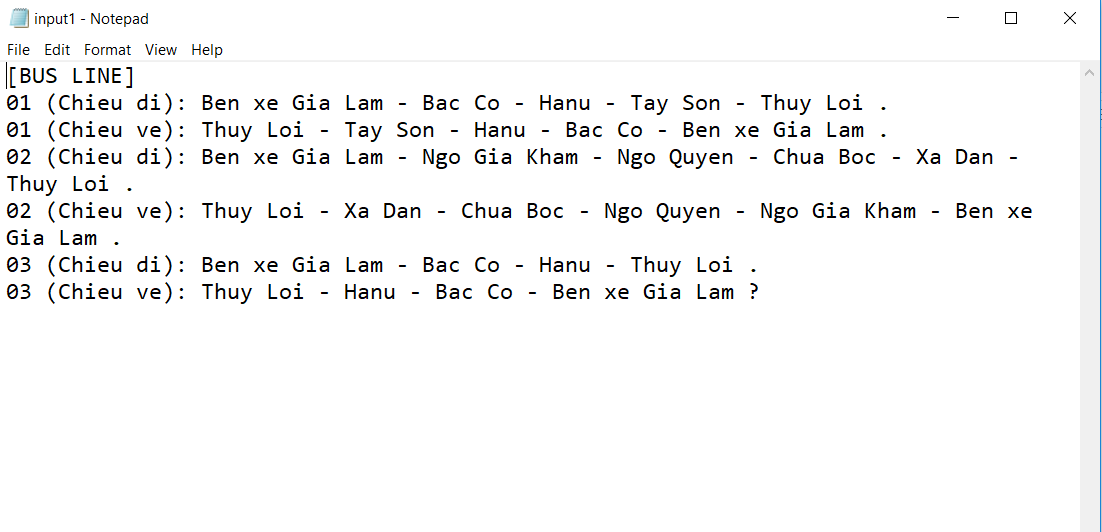


\_ Kiểm thử thông tin của Bến xe Gia Lâm

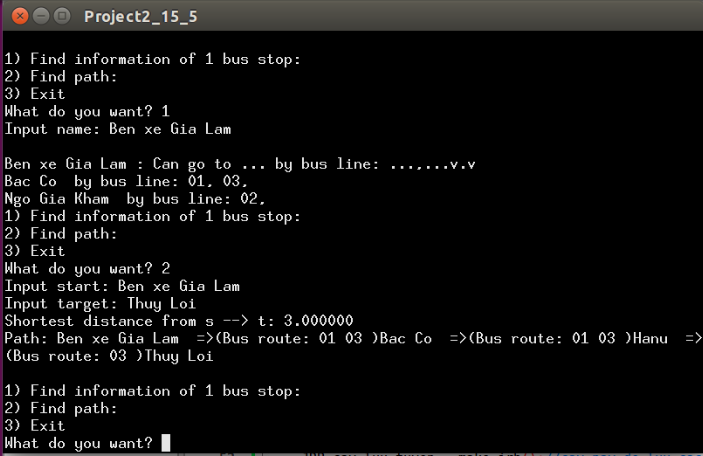


## 2. Tìm kiếm đường đi ngắn nhất

\_ Kiểm thử độ chính xác với dữ liệu nhỏ:



\_Tìm đường đi ngắn nhất giữa Bến xe Gia Lâm và Thủy Lợi



## 3. Demo tìm đường đi với các tuyến Bus thực tế

\_ Dữ liệu bản đồ bus thực tế với 3 xe 01, 02, 03:



\_ Tìm đường đi ngắn nhất giữa Bến xe Gia Lâm và Lê Duẩn

