ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN THỰC HÀNH CUỐI KỲ

MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT
NGƯỜI THỰC HIỆN: Võ Tấn Đạt
MSSV: 1712336

❖Khóa 2017 - 2021❖

MỤC LỤC

I. BÀI TOÁN

Hãng hàng không HPAir (High planes Airline Company) muốn xây dựng một chương trình giúp cho khách hàng có thể tra cứu thông tin chuyến bay. Khách hàng sẽ nhập vào điểm khởi hành, điểm đến, và loại ưu tiên tìm kiếm: chi phí hay tổng thời gian bay. Chương trình sẽ xuất ra **K lộ trình bay tốt nhất** (chi phí / tổng thời gian nhỏ nhất) cho khách hàng và sắp xếp theo thứ tự ưu tiên giảm dần.

Cách thức:

Khi đã **khởi tạo** xong **đồ thị** (bài toán này dùng dach sách kề là hợp lý nhất)

Khởi tạo 1 **hàng đợi** trống và chèn tất cả **đường dẫn** của **đỉnh** xuất phát vào hàng đợi

Trong khi **hàng đợi** không trống và **số đường đi** đã thực hiện < **K** đường đi yêu cầu thì:

- Lấy **đường đi ngắn nhất** trong hàng đợi ra khỏi
- Nếu **điểm cuối** của đường đi vừa lấy ra có **trùng** với **điểm kết thúc** thì **thêm** đường đi này vào danh sách kết quả và tiếp tục thực hiện vòng lặp mới (**continue**)
- Còn không **xét** các **đỉnh kề** của nó, nếu chúng không thuộc đường đi hiện tại thì tạo một **đường dẫn mới** với chi phí bằng cạnh của chúng cộng thêm cho đường đi ngắn nhất vừa lấy ra và **thêm** chúng vào **hàng đợi**.

II. CẤU TRÚC CHƯƠNG TRÌNH

```
void Doc_DuLieu();
```

- Đọc tên của các địa điểm có phục vụ của hãng máy bay và thêm vào danh sách **TPho**
- Đọc các chuyến đi của hãng bay, sau đó tìm các thành phố trong danh sách **TPho** liên kết chúng lại vs nhau

```
Duong_Di DDi_min(vector<Duong_Di>& paths, int DK);
```

- Hàm đơn giản tìm đường đi có chi phí nhỏ nhất với điều kiện truyền vào nếu **1 là so sánh theo giá** nếu **2 so sánh theo thời gian**

```
void Xuly(YeuCau request, ostream& out)
{
	T_Pho* D_BDau = TH_CoTen(request.TH_Di);
	T_Pho* D_KThuc = TH_CoTen(request.TH_Den);

	vector<Duong_Di> result;
	if (D_BDau && D_KThuc)
	{
		// Dùng vector thay thế cho hàng đợi nhưng các thao tác không thay đổi
		// pop push top
		vector<Duong_Di> paths;

		//Thêm tất cả các đỉnh kề đỉnh xuất phát vào hàng đợi
		for (int i = 0; i < D_BDau->LT_Bay.size(); ++i)
		{
			Duong_Di D_Dan;
			D_Dan.DBay.push_back(D_BDau->LT_Bay[i]);
			D_Dan.T_Gia = D_BDau->LT_Bay[i]->Gia;
			D_Dan.T_TGian = D_BDau->LT_Bay[i]->TGian;
			paths.push_back(D_Dan);
		}

		while (!paths.empty() && result.size() < request.k_DDi)
		{
			// Lấy đường đi có tổng chi phí ngắn nhất ra khỏi hàng đợi (pop)
			Duong_Di path = DDi_min(paths, request.DK);

			// Xét đỉnh cuối của đường đi
			// Nếu là đỉnh kết thúc thì thêm đường đi này vào danh sách kết
			// quả và bắt đầu vòng lặp khác
			T_Pho* last = path.DBay.back()->D_KThuc;
			if (last == D_KThuc) {
				result.push_back(path);
				continue;
			}
			//Ngược lại, xét các đỉnh kề của nó, nếu chúng
			//không thuộc đường đi hiện tại thì thêm chúng vào
			//hàng đợi như là một đường đi mới và cập nhật chi
			//phí cho đường đi này

			for (int i = 0; i < last->LT_Bay.size(); i++) {
				T_Pho* next = last->LT_Bay[i]->D_KThuc;
				if (!path.KT_Trung(next)) {
```

```
                                Duong_Di D_Dan = path;
                                D_Dan.DBay.push_back(last->LT_Bay[i]);
                                D_Dan.T_Gia += last->LT_Bay[i]->Gia;
                                D_Dan.T_TGian += last->LT_Bay[i]->TGian;
                                paths.push_back(D_Dan);
                            }
                        }
                    }
                }
        // Xuất kết quả
        xuat_KQ(request, result, out);
}
```

```
void xuat_KQ(YeuCau request, vector<Duong_Di>& result, ostream & out);
```

- Kiểm tra xem các thành phố yêu cầu có trong danh sách kết quả đường đi hay không và thông báo
- In ra các thông tin cho tường đường đi

```
void Xuat_TTin();
```

- Đọc file yêu cầu về số đường đi cần xuất ra
- Đọc lần lược các cặp thành phố mà người dùng muốn truy cập thông tin chuyến bay sau đó gọi hàm `void Xuly(YeuCau request, ostream& out)` để xử lý và in ra file.

III. KIỂM THỬ CHƯƠNG TRÌNH

- **Khởi tạo:**

  Ha Noi,Da Nang 15 75 2
  Da Nang,Can Tho 17 80 3
  Ha Noi,Can Tho 35 190 4

- **Yêu cầu:**

  2
  Ha Noi, Can Tho 1
  Ha Noi, Sai Gon 2

- **Kết quả:**

```
Request is to fly from Ha Noi to Can Tho

Route 1: Ha Noi - Da Nang - Can Tho
Flight #15 from Ha Noi to Da Nang
        Cost:    $75
        Duration: $2 hour(s)
Flight #17 from Da Nang to Can Tho
        Cost:    $80
```

```
        Duration: $3 hour(s)
Total Gia.....$155
Total TGian..... 5 hour(s)
Route 2: Ha Noi - Can Tho
Flight #35 from Ha Noi to Can Tho
        Cost:    $190
        Duration: $4 hour(s)
Total Gia.....$190
Total TGian..... 4 hour(s)


Request is to fly from Ha Noi to Sai Gon

Sorry.HPAir does not serve
Ha Noi!
```

Mức độ hoàn thành: 100%

IV. NGUỒN THAM KHẢO

https://en.wikipedia.org/wiki/K_shortest_path_routing?fbclid=IwAR2K-oJmkuLO68B3wweUY2XKMY6N1_xxFmnlJku0RxKk5IkEgG_GUTwznqM