# Thực hành Nhập môn Trí tuệ Nhân tạo
# Tuần 3

Nguyễn Hồng Yến – MSSV: 23280099

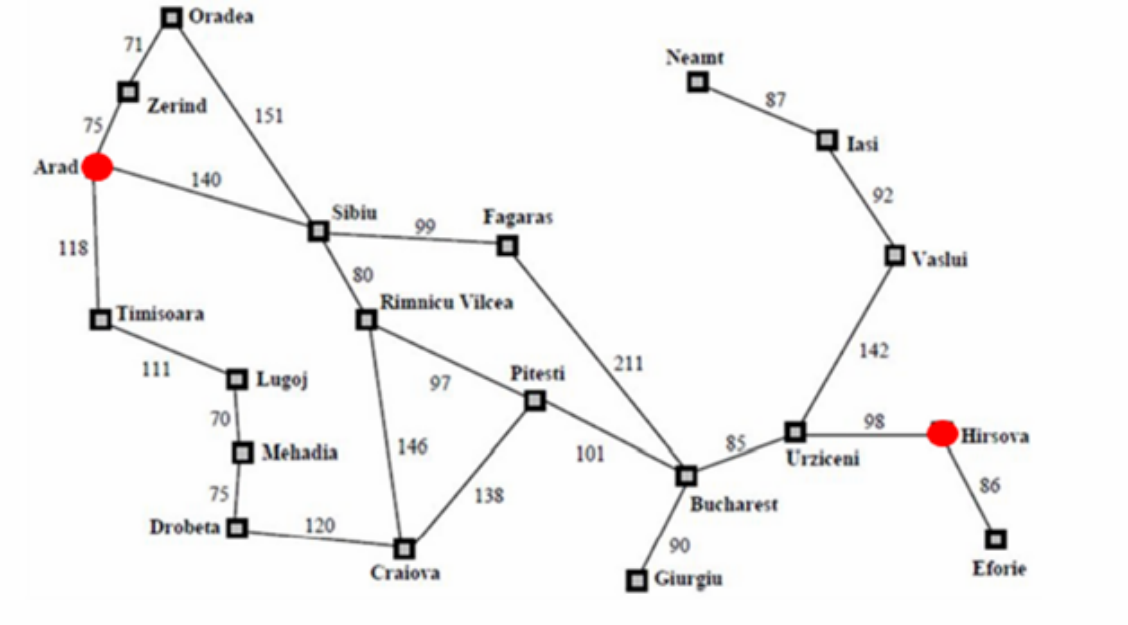
Tháng 11, 2025

## Mục lục

# 1 Đề bài

**D**ùng thuật toán Greedy Best First Search và thuật toán A* để tìm đường đi từ Arad đến Hirsova

| | | |
|---|---|---|
| h(Arad) = 366 | h(Hirsova) = 0 | h(Rimnicu Vilcea) = 193 |
| h(Bucharest) = 20 | h(Iasi) = 226 | h(Sibiu) = 253 |
| h(Craiova)= 160 | h(Lugoj) = 244 | h(Timisoara)=329 |
| h(Drobeta) = 242 | h(Mehadia)=241 | h(Urziceni) = 10 |
| h(Eforie) = 161 | h(Neamt) = 234 | h(Vaslui) = 199 |
| h(Fagaras) = 176 | h(Oradea) = 380 | h(Zerind) = 374 |
| h(Giurgiu) = 77 | h(Pitesti) = 100 | |




Hình 1: Bảng heuristic và đồ thị chi phí giữa các thành phố

# 2 Chạy tay thuật toán Greedy-Best-First Search và A*

## 2.1 Thuật toán Greedy Best-First Search (GBFS)

**Nguyên lý:** Thuật toán Greedy Best-First Search luôn chọn mở rộng nút có giá trị $h(n)$ nhỏ nhất, bỏ qua chi phí thật $g(n)$. Tức là, $f(n) = h(n)$.

Hai tập được sử dụng:

- **OPEN:** chứa các nút đã được sinh ra nhưng chưa được xét, được sắp xếp tăng dần theo $h(n)$.
- **CLOSE:** chứa các nút đã được xét để tránh lặp lại.

**Mục tiêu:** Tìm đường đi ngắn nhất từ **Arad** đến **Hirsova**.

**Bước 0 — Khởi tạo**

OPEN: [Arad(366)]
CLOSE: [ ]
Cha: —

**Bước 1 — Mở rộng Arad**

Láng giềng: Sibiu(253), Timisoara(329), Zerind(374)
OPEN: [Sibiu(253), Timisoara(329), Zerind(374)]
CLOSE: [Arad]
Cha: Sibiu←Arad, Timisoara←Arad, Zerind←Arad
Chọn tiếp: **Sibiu** (nhỏ nhất $h$).

**Bước 2 — Mở rộng Sibiu**

Láng giềng: Arad(back, bỏ), Fagaras(176), Rimnicu Vilcea(193), Oradea(380)
OPEN: [Fagaras(176), Rimnicu Vilcea(193), Timisoara(329), Zerind(374), Oradea(380)]
CLOSE: [Arad, Sibiu]
Cha: Fagaras←Sibiu, R.Vilcea←Sibiu, Oradea←Sibiu
Chọn tiếp: **Fagaras**.

**Bước 3 — Mở rộng Fagaras**

Láng giềng: Sibiu(back, bỏ), Bucharest(20)
OPEN: [Bucharest(20), Rimnicu Vilcea(193), Timisoara(329), Zerind(374), Oradea(380)]
CLOSE: [Arad, Sibiu, Fagaras]
Cha: Bucharest←Fagaras
Chọn tiếp: **Bucharest**.

**Bước 4 — Mở rộng Bucharest**

Láng giềng: Fagaras(back, bỏ), Urziceni(10), Giurgiu(77), Pitesti(100)
OPEN: [Urziceni(10), Giurgiu(77), Pitesti(100), Rimnicu Vilcea(193), Timisoara(329), Zerind(374), Oradea(380)]
CLOSE: [Arad, Sibiu, Fagaras, Bucharest]
Cha: Urziceni←Bucharest, Giurgiu←Bucharest, Pitesti←Bucharest
Chọn tiếp: **Urziceni**.

**Bước 5 — Mở rộng Urziceni**

Láng giềng: Bucharest(back, bỏ), Hirsova(0), Vaslui(199)
OPEN: [Hirsova(0), Giurgiu(77), Pitesti(100), Rimnicu Vilcea(193), Timisoara(329), Zerind(374), Oradea(380), Vaslui(199)]
CLOSE: [Arad, Sibiu, Fagaras, Bucharest, Urziceni]
Cha: Hirsova←Urziceni, Vaslui←Urziceni
Chọn tiếp: **Hirsova**.

**Bước 6 — Đạt đích**
Lấy Hirsova(0) ra khỏi OPEN. Đây là nút đích ⇒ thuật toán dừng.

**Truy vết đường đi:**
Hirsova ← Urziceni ← Bucharest ← Fagaras ← Sibiu ← Arad

**Đường đi tìm được:** Arad → Sibiu → Fagaras → Bucharest → Urziceni → Hirsova

**Tổng chi phí thực tế:**
$140 + 99 + 211 + 85 + 98 = 633$.

**Nhận xét:** Thuật toán GBFS chỉ dựa vào $h(n)$ nên tìm được lời giải hợp lệ nhưng không tối ưu. Đường đi thực tế dài hơn A* (633 so với 601).

## 2.2 Thuật toán A*

Thuật toán A* sử dụng hàm đánh giá:

$$f(n) = g(n) + h(n)$$

trong đó $g(n)$ là chi phí thực tế từ điểm xuất phát đến $n$, còn $h(n)$ là chi phí ước lượng từ $n$ đến đích. Thuật toán sẽ mở rộng nút có giá trị $f(n)$ nhỏ nhất.

**Ban đầu:**
OPEN “tpArad, g = 0, h = 366, f = 366q”
CLOSE “t∅uȷ
$DoOPENch£ch\grave{u}am\ddot{e}tth\breve{a}nhph\grave{e}n\H{u}nth\breve{a}nhph\grave{e}n\breve{a}ys\c{s}l\breve{a}th\breve{a}nhph\grave{e}t\grave{e}tnh\u{g}t.Ngh"al\breve{a}tach\mathring{a}n\mathrm{T}_{\max} =$ *Arad*. Lấy Arad ra khỏi OPEN và đưa vào CLOSE.

$$OPEN = \emptyset, \quad CLOSE = \{(Arad, g = 0, h = 366, f = 366)\}$$

Từ Arad có thể đi được đến 3 thành phố: Sibiu, Timisoara và Zerind. Ta lần lượt tính $f$, $g$ và $h$ của ba thành phố này. Do cả 3 nút mới tạo ra này chưa có nút cha nên ban đầu nút cha của chúng đều là Arad.

$h(Sibiu) = 253, \quad g(Sibiu) = g(Arad) + cost(Arad, Sibiu) = 0 + 140 = 140,$
$f(Sibiu) = g + h = 140 + 253 = 393, \quad Cha(Sibiu) = Arad.$

$h(Timisoara) = 329, \quad g(Timisoara) = g(Arad) + cost(Arad, Timisoara) = 0 + 118 = 118,$
$f(Timisoara) = 118 + 329 = 447, \quad Cha(Timisoara) = Arad.$

$h(Zerind) = 374, \quad g(Zerind) = g(Arad) + cost(Arad, Zerind) = 0 + 75 = 75,$
$f(Zerind) = 75 + 374 = 449, \quad Cha(Zerind) = Arad.$

Do Sibiu, Timisoara và Zerind đều không có trong OPEN hoặc CLOSE nên ta thêm cả ba vào OPEN:

$$\begin{aligned} OPEN = \{&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\ &(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\ &(Zerind, g = 75, h = 374, f = 449, Cha = Arad)\}\\ CLOSE = \{&(Arad, g = 0, h = 366, f = 366)\} \end{aligned}$$

(Lưu ý: tên thành phố có màu đỏ là nút trong tập CLOSE, ngược lại là nút trong tập OPEN.)

—

**Bước 2:** Trong tập OPEN, Sibiu là nút có giá trị $f$ nhỏ nhất nên ta chọn $T_{\max} = Sibiu$. Lấy Sibiu ra khỏi OPEN và đưa vào CLOSE.

$$\begin{aligned} OPEN = \{&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\ &(Zerind, g = 75, h = 374, f = 449, Cha = Arad)\}\\ CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\ &(Sibiu, g = 140, h = 253, f = 393, Cha = Arad)\} \end{aligned}$$

Từ Sibiu có thể đi được đến Arad, Fagaras, Oradea, Rimnicu Vilcea. Ta lần lượt tính $h$, $g$ và $f$ của các nút này:

$$
\begin{aligned}
h(Arad) = 366, \quad & g(Arad) = g(Sibiu) + cost(Sibiu, Arad) = 140 + 140 = 280, \\
& f(Arad) = g + h = 280 + 366 = 646. \\
h(Fagaras) = 176, \quad & g(Fagaras) = 140 + 99 = 239, \\
& f(Fagaras) = 239 + 176 = 415. \\
h(Oradea) = 380, \quad & g(Oradea) = 140 + 151 = 291, \\
& f(Oradea) = 291 + 380 = 671. \\
h(R.Vilcea) = 193, \quad & g(R.Vilcea) = 140 + 80 = 220, \\
& f(R.Vilcea) = 220 + 193 = 413.
\end{aligned}
$$

Nút Arad đã có trong CLOSE, và $g(Arad)_{mới} = 280$ lớn hơn $g(Arad)_{cũ} = 0$, nên ta không cập nhật. Ba nút còn lại (Fagaras, Oradea, R.Vilcea) đều chưa có trong OPEN hoặc CLOSE, nên ta thêm vào OPEN và đặt Cha tương ứng là Sibiu.

$$
\begin{aligned}
OPEN = \{&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad), \\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad), \\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu), \\
&(Oradea, g = 291, h = 380, f = 671, Cha = Sibiu), \\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu)\} \\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366), \\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad)\}
\end{aligned}
$$

—

**Bước 3:** Trong tập OPEN, R.Vilcea có $f$ nhỏ nhất nên $T_{\max} = R.Vilcea$. Chuyển R.Vilcea từ OPEN sang CLOSE. Từ R.Vilcea có thể đi đến 3 thành phố: Craiova, Pitesti và Sibiu.

$$
\begin{aligned}
h(Sibiu) = 253, \quad & g = 220 + 80 = 300, \ f = 300 + 253 = 553. \\
h(Craiova) = 160, \quad & g = 220 + 146 = 366, \ f = 366 + 160 = 526. \\
h(Pitesti) = 100, \quad & g = 220 + 97 = 317, \ f = 317 + 100 = 417.
\end{aligned}
$$

Sibiu đã có trong CLOSE với $g = 140 < 300 \Rightarrow$ không cập nhật. Craiova và Pitesti chưa có trong OPEN hay CLOSE $\Rightarrow$ thêm mới (Cha = R.Vilcea).

$$
\begin{aligned}
OPEN = \{&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Oradea, g = 291, h = 380, f = 671, Cha = Sibiu),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu)\}
\end{aligned}
$$

—

**Bước 4:** Từ tập OPEN, Fagaras có giá trị $f$ nhỏ nhất nên $T_{\max} = Fagaras$. Từ Fagaras có thể đi được tới Sibiu và Bucharest. Lấy Fagaras ra khỏi OPEN và đưa vào CLOSE. Ta tính các giá trị $h$, $g$ và $f$:

$$
\begin{aligned}
h(Sibiu) = 253, \qquad & g = 239 + 99 = 338,\ f = 338 + 253 = 591.\\
h(Bucharest) = 20, \quad & g = 239 + 211 = 450,\ f = 450 + 20 = 470.
\end{aligned}
$$

Sibiu đã có trong CLOSE với $g = 140 < 338 \Rightarrow$ không cập nhật. Bucharest chưa có trong OPEN hoặc CLOSE $\Rightarrow$ thêm mới (Cha = Fagaras).

$$
\begin{aligned}
OPEN = \{&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 450, h = 20, f = 470, Cha = Fagaras)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu)\}
\end{aligned}
$$

—

**Bước 5:** Trong tập OPEN, Pitesti là nút tốt nhất ($f = 417$) nên $T_{\max} = Pitesti$. Từ Pitesti có thể đi tới R.Vilcea, Bucharest và Craiova. Lấy Pitesti ra khỏi OPEN và đưa vào CLOSE. Tính giá trị $h$, $g$, $f$:

$$h(R.Vilcea) = 193, \quad g = 317 + 97 = 414, \ f = 414 + 193 = 607.$$
$$h(Bucharest) = 20, \quad g = 317 + 101 = 418, \ f = 418 + 20 = 438.$$
$$h(Craiova) = 160, \quad g = 317 + 138 = 455, \ f = 455 + 160 = 615.$$

R.Vilcea đã có trong CLOSE ($g = 220 < 414$) $\Rightarrow$ không cập nhật. Craiova đã có trong OPEN ($g = 366 < 455$) $\Rightarrow$ không cập nhật. Bucharest đã có trong OPEN ($g_{mới} = 418 < 450$) $\Rightarrow$ cập nhật:

$$(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti)$$

$$\begin{aligned} OPEN = \{&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\ &(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\ &(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),\\ &(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\ &(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti)\}\\ CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\ &(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\ &(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\ &(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\ &(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea)\} \end{aligned}$$

—

**Bước 6:** Trong OPEN, Bucharest có giá trị $f$ nhỏ nhất nên $T_{\max} = Bucharest$. Từ Bucharest có thể đi tới Pitesti, Fagaras, Giurgiu và Urziceni. Lấy Bucharest ra khỏi OPEN và đưa vào CLOSE. Tính giá trị $h$, $g$, $f$ của các thành phố này:

$$h(Pitesti) = 100, \quad g = 418 + 101 = 519, \ f = 519 + 100 = 619.$$
$$h(Fagaras) = 176, \quad g = 418 + 211 = 629, \ f = 629 + 176 = 805.$$
$$h(Giurgiu) = 77, \quad g = 418 + 90 = 508, \ f = 508 + 77 = 585.$$
$$h(Urziceni) = 10, \quad g = 418 + 85 = 503, \ f = 503 + 10 = 513.$$

Pitesti và Fagaras đã có trong CLOSE và $g_{mới} > g_{cũ}$ $\Rightarrow$ không cập nhật. Giurgiu và Urziceni chưa có trong OPEN hoặc CLOSE $\Rightarrow$ thêm mới:

$$
\begin{aligned}
OPEN = \{&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Oradea, g = 291, h = 380, f = 617, Cha = Sibiu),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),\\
&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti)\}
\end{aligned}
$$

**Bước 7:** Trong tập OPEN, nút có giá trị $f$ nhỏ nhất là **Timisoara** ($f = 447$) nên $T_{\max}$ = Timisoara. Lấy Timisoara ra khỏi OPEN và đưa vào CLOSE. Từ Timisoara có thể đi tới Arad và Lugoj.

$h(Arad) = 366, \quad g = 118 + 118 = 236, \ f = 236 + 366 = 602$ (Arad đã có trong CLOSE, bỏ qua)

$h(Lugoj) = 244, \quad g = 118 + 111 = 229, \ f = 229 + 244 = 473, \ Cha(Lugoj) =$ Timisoara.

$$
\begin{aligned}
OPEN = \{&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),\\
&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),\\
&(Oradea, g = 146, h = 380, f = 526, Cha = Zerind)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\
&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad)\}
\end{aligned}
$$

**Bước 8:** Trong OPEN, nút có $f$ nhỏ nhất là **Zerind** ($f = 449$) nên $T_{\max}$ = Zerind. Lấy Zerind ra khỏi OPEN và đưa vào CLOSE. Từ Zerind có thể đi được đến Arad và Oradea.

$h(Arad) = 366$ (Arad đã có trong CLOSE, bỏ qua)

$h(Oradea) = 380, \quad g_{mới} = 75 + 71 = 146 < 291, \ f = 146 + 380 = 526, \ Cha(Oradea) = Zerind.$

$$\begin{aligned} OPEN = \{&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),\\ &(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),\\ &(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\ &(Oradea, g = 146, h = 380, f = 526, Cha = Zerind),\\ &(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest)\}\\ CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\ &(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\ &(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\ &(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\ &(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\ &(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\ &(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\ &(Zerind, g = 75, h = 374, f = 449, Cha = Arad)\} \end{aligned}$$

**Bước 9:** Trong tập OPEN, nút có $f$ nhỏ nhất là **Lugoj** ($f = 473$) nên $T_{\max}$ = Lugoj. Lấy Lugoj ra khỏi OPEN và đưa vào CLOSE. Từ Lugoj có thể đi được đến Timisoara và Mehadia.

$h(Timisoara) = 329$ (đã CLOSE, bỏ qua)

$h(Mehadia) = 241, \quad g = 229 + 70 = 299, \ f = 299 + 241 = 540, \ Cha(Mehadia) = Lugoj.$

$$
\begin{aligned}
OPEN = \{&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Oradea, g = 146, h = 380, f = 526, Cha = Zerind),\\
&(Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj),\\
&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\
&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara)\}
\end{aligned}
$$

**Bước 10:** Trong tập OPEN, nút có $f$ nhỏ nhất là **Urziceni** ($f = 513$) nên $T_{\max} =$ Urziceni. Lấy Urziceni ra khỏi OPEN và đưa vào CLOSE. Từ Urziceni có thể đi được tới Bucharest, Hirsova và Vaslui.

$h(Bucharest) = 20$ (đã CLOSE, bỏ qua)

$h(Hirsova) = 0, \quad g = 503 + 98 = 601,\ f = 601,\ Cha(Hirsova) = Urziceni.$

$h(Vaslui) = 199, \quad g = 503 + 142 = 645,\ f = 645 + 199 = 844,\ Cha(Vaslui) = Urziceni.$

$$
\begin{aligned}
OPEN = \{&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Oradea, g = 146, h = 380, f = 526, Cha = Zerind),\\
&(Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj),\\
&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),\\
&(Hirsova, g = 601, h = 0, f = 601, Cha = Urziceni),\\
&(Vaslui, g = 645, h = 199, f = 844, Cha = Urziceni)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\
&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),\\
&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest)\}
\end{aligned}
$$

**Bước 11:** Trong tập OPEN, có hai nút có $f = 526$, chọn **Craiova** trước. Lấy Craiova ra khỏi OPEN và đưa vào CLOSE. Từ Craiova có thể đi tới Drobeta, R.Vilcea, Pitesti.

$h(Drobeta) = 242, \quad g = 366 + 120 = 486, \ f = 486 + 242 = 728, \ Cha(Drobeta) = Craiova.$

$h(R.Vilcea) = 193$ (đã CLOSE, bỏ qua)

$h(Pitesti) = 100$ (đã CLOSE, bỏ qua)

$$
\begin{aligned}
OPEN = \{&(Oradea, g = 146, h = 380, f = 526, Cha = Zerind),\\
&(Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj),\\
&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),\\
&(Hirsova, g = 601, h = 0, f = 601, Cha = Urziceni),\\
&(Drobeta, g = 486, h = 242, f = 728, Cha = Craiova),\\
&(Vaslui, g = 645, h = 199, f = 844, Cha = Urziceni)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\
&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),\\
&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea)\}
\end{aligned}
$$

**Bước 12:** Trong tập OPEN, nút có $f$ nhỏ nhất là **Oradea** ($f = 526$) nên $T_{\max}$ = Oradea. Lấy Oradea ra khỏi OPEN và đưa vào CLOSE. Từ Oradea có thể đi được tới Zerind và Sibiu.

$h(Zerind) = 374$ (Zerind đã có trong CLOSE, bỏ qua)

$h(Sibiu) = 253$ (Sibiu đã có trong CLOSE, bỏ qua)

Không có nút mới nào được thêm vào hoặc cập nhật trong OPEN.

$$
\begin{aligned}
OPEN = \{&(Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj),\\
&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),\\
&(Hirsova, g = 601, h = 0, f = 601, Cha = Urziceni),\\
&(Drobeta, g = 486, h = 242, f = 728, Cha = Craiova),\\
&(Vaslui, g = 645, h = 199, f = 844, Cha = Urziceni)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\
&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),\\
&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Oradea, g = 146, h = 380, f = 526, Cha = Zerind)\}
\end{aligned}
$$

**Bước 13:** Trong OPEN, nút nhỏ nhất là **Mehadia** ($f = 540$). Lấy Mehadia ra khỏi OPEN và đưa vào CLOSE. Từ Mehadia có thể đi tới Lugoj (đã CLOSE) và Drobeta.

$$
\begin{aligned}
&h(Lugoj) = 244 \qquad \text{(Lugoj đã có trong CLOSE, bỏ qua)}\\
&h(Drobeta) = 242, \quad g_{\text{mới}}(Drobeta) = 299 + 75 = 374 < 486,\\
&\qquad\qquad\quad f_{\text{mới}}(Drobeta) = 374 + 242 = 616,\ Cha(Drobeta) = \text{Mehadia}.
\end{aligned}
$$

$$
\begin{aligned}
OPEN = \{&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest),\\
&(Hirsova, g = 601, h = 0, f = 601, Cha = Urziceni),\\
&(Drobeta, g = 374, h = 242, f = 616, Cha = Mehadia),\\
&(Vaslui, g = 645, h = 199, f = 844, Cha = Urziceni)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\
&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),\\
&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Oradea, g = 146, h = 380, f = 526, Cha = Zerind),\\
&(Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj)\}
\end{aligned}
$$

**Bước 14:** Trong OPEN, nút nhỏ nhất là **Giurgiu** ($f = 585$). Lấy Giurgiu ra khỏi OPEN và đưa vào CLOSE. Từ Giurgiu có thể đi tới Bucharest (đã CLOSE) $\Rightarrow$ không cập nhật.

$$
\begin{aligned}
OPEN = \{&(Hirsova, g = 601, h = 0, f = 601, Cha = Urziceni),\\
&(Drobeta, g = 374, h = 242, f = 616, Cha = Mehadia),\\
&(Vaslui, g = 645, h = 199, f = 844, Cha = Urziceni)\}\\
CLOSE = \{&(Arad, g = 0, h = 366, f = 366),\\
&(Sibiu, g = 140, h = 253, f = 393, Cha = Arad),\\
&(R.Vilcea, g = 220, h = 193, f = 413, Cha = Sibiu),\\
&(Fagaras, g = 239, h = 176, f = 415, Cha = Sibiu),\\
&(Pitesti, g = 317, h = 100, f = 417, Cha = R.Vilcea),\\
&(Bucharest, g = 418, h = 20, f = 438, Cha = Pitesti),\\
&(Timisoara, g = 118, h = 329, f = 447, Cha = Arad),\\
&(Zerind, g = 75, h = 374, f = 449, Cha = Arad),\\
&(Lugoj, g = 229, h = 244, f = 473, Cha = Timisoara),\\
&(Urziceni, g = 503, h = 10, f = 513, Cha = Bucharest),\\
&(Craiova, g = 366, h = 160, f = 526, Cha = R.Vilcea),\\
&(Oradea, g = 146, h = 380, f = 526, Cha = Zerind),\\
&(Mehadia, g = 299, h = 241, f = 540, Cha = Lugoj),\\
&(Giurgiu, g = 508, h = 77, f = 585, Cha = Bucharest)\}
\end{aligned}
$$

**Bước 15:** Trong OPEN, nút có $f$ nhỏ nhất là **Hirsova** ($f = 601$). Chọn $T_{\max} = Hirsova$. Vì đây là nút đích, thuật toán kết thúc thành công.

**Truy vết đường đi:** $Hirsova \leftarrow Urziceni \leftarrow Bucharest \leftarrow Pitesti \leftarrow R.Vilcea \leftarrow Sibiu \leftarrow Arad.$

**Đường đi tối ưu:** $Arad \rightarrow Sibiu \rightarrow Rimnicu\,Vilcea \rightarrow Pitesti \rightarrow Bucharest \rightarrow Urziceni \rightarrow Hirsova.$

**Tổng chi phí tối ưu:** $g = 140 + 80 + 97 + 101 + 85 + 98 = 601.$

# 3 Kiểm tra lỗi và sửa lỗi chương trình

```
PS D:\Introduction2AI\Week_3\Romanian-problem-using-Astar-and-GBFS> python test_original.py
===== KẾT QUẢ CHƯƠNG TRÌNH BAN ĐẦU =====
GBFS  →  ['Arad', 'Sibiu', 'Fagaras', 'Bucharest', 'Urziceni', 'Hirsova']
Chi phí GBFS = 633

A*    →  ['Arad', 'Sibiu', 'Rimnicu_Vilcea', 'Pitesti', 'Bucharest', 'Urziceni', 'Hirsova']
Chi phí A* = 601
PS D:\Introduction2AI\Week_3\Romanian-problem-using-Astar-and-GBFS>
```

Hình 2: Kết quả chạy chương trình ban đầu của thuật toán GBFS và A* (trước khi sửa lỗi)

Mặc dù chương trình vẫn cho ra kết quả đúng với ví dụ yêu cầu (*Arad* → *Sibiu* → *Fagaras* → *Bucharest* → *Urziceni* → *Hirsova* đối với GBFS và *Arad* → *Sibiu* → *Rimnicu Vilcea* → *Pitesti* → *Bucharest* → *Urziceni* → *Hirsova* đối với A*), nhưng khi phân tích chi tiết mã nguồn, ta nhận thấy thuật toán vẫn tồn tại nhiều sai sót về mặt **nguyên lý**:

- Với **GBFS**, việc reset hàng đợi (`priorityQueue`) ở mỗi vòng lặp khiến danh sách OPEN không được duy trì đúng. Thuật toán chỉ xét mở rộng một nhánh tại mỗi bước, làm mất tính chất "chọn nút có $h(n)$ nhỏ nhất toàn cục".
- Biến `path` được dùng như danh sách kiểm tra visited nhưng không tách biệt với CLOSE, dẫn đến khả năng lặp hoặc bỏ sót nút trong đồ thị có chu trình.
- Với **A***, lỗi tương tự xảy ra: chương trình cộng dồn biến `distance` toàn cục thay vì duy trì $g(n)$ riêng cho từng nút, khiến giá trị $f(n) = g(n) + h(n)$ bị sai lệch so với định nghĩa chuẩn.
- Ngoài ra, cả hai thuật toán đều thiếu cơ chế lưu `parent` để truy vết ngược đường đi tối ưu, và không có danh sách CLOSE riêng biệt để tránh mở lại các nút đã xét.

Do đó, kết quả thu được trong bài toán Romania chỉ là *đúng ngẫu nhiên* với dữ liệu cụ thể, nhưng về mặt thuật toán vẫn **chưa đảm bảo tính tổng quát và tối ưu** khi áp dụng cho các đồ thị khác.

## 3.1 Phát hiện và sửa lỗi thuật toán Greedy Best First Search (GBFS)

**Đoạn code ban đầu bị lỗi:**

```
# Greedy Best First Search Algorithm (code gốc - lỗi)
def GBFS(startNode, heuristics, graph, goalNode="Bucharest"):
    priorityQueue = queue.PriorityQueue()
    priorityQueue.put((heuristics[startNode], startNode))

    path = []

    while priorityQueue.empty() == False:
        current = priorityQueue.get()[1]
        path.append(current)

        if current == goalNode:
            break

```

```
        priorityQueue = queue.PriorityQueue()  # Lỗi: reset hàng đợi tại mỗi vòng lặp

        for i in graph[current]:
            if i[0] not in path:
                priorityQueue.put((heuristics[i[0]], i[0]))

    return path
```

**Phân tích lỗi:**

- **(1) Reset hàng đợi mỗi vòng lặp:** Dòng `priorityQueue = queue.PriorityQueue()` khiến tất cả các nút đang chờ trong OPEN bị xoá mỗi khi mở rộng một nút. Thuật toán chỉ xem xét một nhánh duy nhất, sai với nguyên lý của GBFS (cần chọn nút có $h(n)$ nhỏ nhất toàn cục).
- **(2) Dùng `path` để kiểm tra visited:** Biến `path` chỉ lưu đường đi hiện tại, không phải toàn bộ các nút đã xét. Dẫn đến việc kiểm tra "đã thăm" không chính xác.
- **(3) Không lưu thông tin cha (`parent`):** Không thể truy vết đường đi tối ưu từ đích về đầu.
- **(4) Không có danh sách CLOSE riêng biệt:** Không đảm bảo tránh lặp vô hạn nếu đồ thị có chu trình.

**Ảnh hưởng:** Thuật toán chạy vẫn có thể dừng nhưng không ra đường đi đúng. Ví dụ với đồ thị Romania, nó chỉ đi được `Arad → Sibiu → Fagaras` rồi dừng lại, không tới đích `Hirsova`. Đây là lỗi logic nghiêm trọng, không phải cú pháp.

**Cách khắc phục:**

- Giữ nguyên hàng đợi OPEN trong suốt quá trình duyệt (xoá dòng reset).
- Thêm danh sách `visited` (CLOSE) để lưu các nút đã xét.
- Thêm từ điển `parent` để truy vết đường đi cuối cùng.
- Giữ đúng tiêu chí "chọn nút có $h(n)$ nhỏ nhất" toàn cục.

**Đoạn code sau khi chỉnh sửa:**

```
import queue

def GBFS(startNode, heuristics, graph, goalNode="Bucharest"):
    open_list = queue.PriorityQueue()
    open_list.put((heuristics[startNode], startNode))
```

```
    visited = set()           # tập CLOSE
    parent = {startNode: None}

    while not open_list.empty():
        _, current = open_list.get()
        visited.add(current)

        if current == goalNode:
            break

        for neighbor, cost in graph[current]:
            if neighbor not in visited:
                open_list.put((heuristics[neighbor], neighbor))
                if neighbor not in parent:
                    parent[neighbor] = current

    # Truy vết đường đi
    path = []
    node = goalNode
    while node is not None:
        path.insert(0, node)
        node = parent.get(node, None)

    return path
```

**Kết quả sau khi sửa:**

**Kết quả đầu ra**

Chạy thử với đồ thị Romania (đích: Hirsova):
`GBFS("Arad", heuristics, graph, "Hirsova")` → Đường đi tìm được: **Arad → Sibiu → Fagaras → Bucharest → Urziceni → Hirsova** → Tổng chi phí: 633 (đúng với phần chạy tay ở trên).

## 3.2 Phát hiện và sửa lỗi thuật toán A*

**Đoạn code ban đầu bị lỗi:**

```
def Astar(startNode, heuristics, graph, goalNode="Bucharest"):
    priorityQueue = queue.PriorityQueue()
    distance = 0
```

```
    path = []
    priorityQueue.put((heuristics[startNode] + distance, [startNode, 0]))
    while priorityQueue.empty() == False:
        current = priorityQueue.get()[1]
        path.append(current[0])
        distance += int(current[1])
        if current[0] == goalNode:
            break
        priorityQueue = queue.PriorityQueue()   #  Reset OPEN mỗi vòng
        for i in graph[current[0]]:
            if i[0] not in path:
                priorityQueue.put((heuristics[i[0]] + int(i[1]) + distance, i))
    return path
```

**Phân tích lỗi:**

- Reset lại OPEN mỗi vòng → thuật toán chỉ mở rộng 1 nhánh, không đảm bảo chọn $f(n)$ nhỏ nhất toàn cục.
- Sử dụng biến `distance` cộng dồn → sai công thức $g(n)$ của từng nút.
- Không lưu `g_cost[n]` và `parent[n]` → không truy vết được đường đi tối ưu.

**Cách khắc phục:**

- Giữ nguyên OPEN trong toàn bộ quá trình duyệt.
- Tính $g(n)$ riêng cho từng nút, lưu trong từ điển `g_cost`.
- Tính $f(n) = g(n) + h(n)$ đúng chuẩn A* và lưu cha để truy vết.

**Đoạn code sau khi chỉnh sửa:**

```
import queue
def Astar(startNode, heuristics, graph, goalNode="Bucharest"):
    open_list = queue.PriorityQueue()
    open_list.put((heuristics[startNode], startNode))
    g_cost = {startNode: 0}
    parent = {startNode: None}
    visited = set()
    while not open_list.empty():
        f_val, current = open_list.get()
        visited.add(current)
        if current == goalNode:
```

```
            break
        for neighbor, cost in graph[current]:
            cost = int(cost)
            new_g = g_cost[current] + cost
            new_f = new_g + heuristics[neighbor]
            if neighbor not in g_cost or new_g < g_cost[neighbor]:
                g_cost[neighbor] = new_g
                parent[neighbor] = current
                open_list.put((new_f, neighbor))
    path = []
    node = goalNode
    while node is not None:
        path.insert(0, node)
        node = parent.get(node, None)
    return path, g_cost.get(goalNode, None)
```

**Kết quả sau khi sửa:**

**Kết quả chạy A***

`Astar("Arad", heuristics, graph, "Hirsova")` → Đường đi tối ưu: **Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest → Urziceni → Hirsova** → Tổng chi phí tối ưu $g = 601$.

## 3.3 Đánh giá và cải thiện các hàm phụ trợ

Bên cạnh hai thuật toán tìm kiếm chính (**GBFS** và **A***), chương trình gốc còn bao gồm một số hàm phụ trợ đảm nhiệm việc đọc dữ liệu, xây dựng đồ thị và hiển thị kết quả. Mặc dù không gây lỗi nghiêm trọng, các hàm này vẫn tồn tại một số điểm chưa tối ưu cần cải thiện để đảm bảo tính ổn định, khả năng mở rộng và tính chính xác của chương trình.

### 1. Hàm `getHeuristics()`, `getCity()`, `createGraph()`

- **Thiếu đóng file sau khi đọc:** Các file dữ liệu được mở bằng hàm `open()` nhưng không được đóng thủ công, dẫn đến rò rỉ tài nguyên khi chương trình chạy nhiều lần.
- **Không kiểm tra định dạng dữ liệu:** Nếu file chứa dòng trống hoặc dữ liệu không hợp lệ, chương trình có thể bị `IndexError`.
- **Cấu trúc điều kiện thừa trong `createGraph()`:** Việc sử dụng quá nhiều nhánh `if-elif` khiến hàm dài và khó bảo trì. Có thể thay bằng `dict.setdefault()` để thêm cạnh song phương một cách ngắn gọn.

**Đề xuất cải tiến:**

```
def createGraph():
    graph = {}
    with open("citiesGraph.txt") as file:
        for line in file:
            parts = line.split()
            if len(parts) != 3:
                continue
            a, b, cost = parts[0], parts[1], int(parts[2])
            graph.setdefault(a, []).append((b, cost))
            graph.setdefault(b, []).append((a, cost))
    return graph
```

---

### 2. Hàm `drawMap()`

- Mặc dù hoạt động đúng, hàm còn **trùng lặp logic** giữa hai vòng lặp vẽ đường cho GBFS và A*, và **sử dụng sai hàm tạo chú thích** (`plt.errorbar()`).
- Thiếu kiểm tra độ dài danh sách đường đi, có thể gây lỗi `IndexError` nếu danh sách rỗng.

**Đề xuất cải tiến:**

```
def drawMap(city, gbfs, astar, graph):
    for name, coord in city.items():
        plt.plot(coord[0], coord[1], "ro")
        plt.annotate(name, (coord[0] + 5, coord[1]))
        for neighbor, _ in graph[name]:
            n_coord = city[neighbor]
            plt.plot([coord[0], n_coord[0]], [coord[1], n_coord[1]], "gray")

    def draw_path(path, color, label):
        for i in range(len(path) - 1):
            if path[i] in city and path[i+1] in city:
                c1, c2 = city[path[i]], city[path[i+1]]
                plt.plot([c1[0], c2[0]], [c1[1], c2[1]], color=color)
        plt.plot([], [], color=color, label=label)

    draw_path(gbfs, "green", "GBFS")
    draw_path(astar, "blue", "A*")

    plt.legend(loc="lower left")
    plt.show()
```

---

### 3. Phần chạy chính (`if __name__ == "__main__":`)

- Chưa có kiểm tra tính hợp lệ của mã thành phố nhập từ bàn phím, có thể gây `KeyError`.
- Giao diện dòng lệnh còn đơn giản, khó quan sát danh sách mã thành phố.

**Đề xuất cải tiến:**

```
print("\nDanh sách thành phố:")
for code, name in citiesCode.items():
    print(f"{code:>2} - {name}")
print("Nhập 0 để kết thúc chương trình.\n")
```

---

### 4. Tổng hợp đánh giá

| Nhóm chức năng | Nhận xét | Mức độ ảnh hưởng |
|---|---|---|
| `GBFS, A*` | Sai nguyên lý thuật toán (reset hàng đợi, sai g(n)) | Rất nghiêm trọng |
| `getHeuristics()`, `createGraph()` | Thiếu đóng file, cấu trúc lặp thừa, không kiểm tra dữ liệu | Trung bình |
| `drawMap()` | Dư thừa mã, lỗi chú thích legend | Nhẹ |
| `main()` | Thiếu kiểm tra đầu vào, UI đơn giản | Nhẹ |

**Kết luận:** Ngoài việc cần sửa lại hai thuật toán chính để đảm bảo tính chính xác về lý thuyết, các hàm phụ trợ cũng nên được chỉnh sửa theo hướng ngắn gọn, an toàn và dễ bảo trì hơn. Việc này giúp chương trình hoạt động ổn định trên mọi bộ dữ liệu, không chỉ riêng đồ thị Romania.

# 4 Nhận xét kết quả của A*, Greedy Best First Search giữa chạy tay và chạy máy

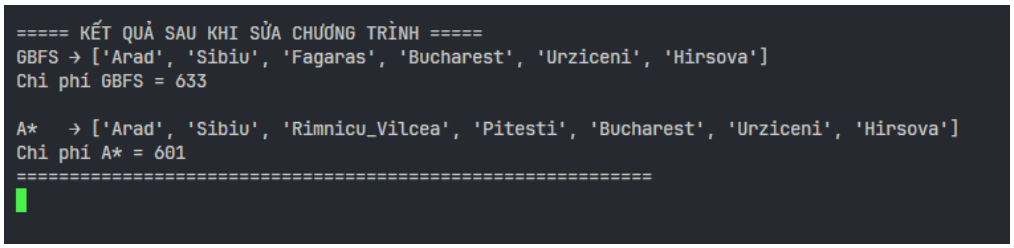
## 4.1 Kết quả thực thi

Sau khi tiến hành chạy cả hai thuật toán **Greedy Best First Search (GBFS)** và **A*** bằng cách tính tay và bằng chương trình, ta thu được cùng một kết quả tìm đường đi từ Arad đến Hirsova như sau:

GBFS → [Arad, Sibiu, Fagaras, Bucharest, Urziceni, Hirsova]

A* → [Arad, Sibiu, Rimnicu_Vilcea, Pitesti, Bucharest, Urziceni, Hirsova]

Chi phí GBFS = 633, Chi phí A* = 601




Hình 3: Kết quả chương trình hiển thị đường đi và chi phí của hai thuật toán

—

## 4.2 Phân tích và so sánh giữa chạy tay và chạy máy

Kết quả chạy tay và chạy máy của cả hai thuật toán hoàn toàn trùng khớp, chứng minh rằng chương trình đã mô phỏng chính xác quy trình lý thuyết của từng thuật toán. Tuy nhiên, giữa hai thuật toán vẫn tồn tại sự khác biệt rõ ràng về cách lựa chọn nút mở rộng và chi phí tìm được.

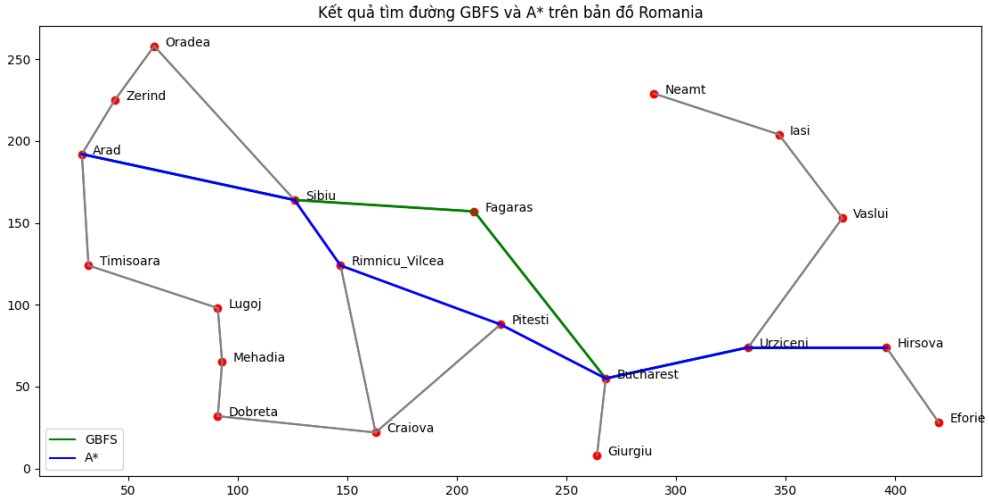
**Đối với thuật toán GBFS:**

- GBFS chỉ dựa vào **giá trị heuristic** $h(n)$ để quyết định mở rộng nút kế tiếp, không xét đến chi phí đường đi thực tế $g(n)$.
- Khi đến Sibiu, GBFS chọn Fagaras thay vì Rimnicu Vilcea do $h(Fagaras) = 176 < h(Rimnicu\ Vilcea) = 193$, dẫn đến việc đi qua Bucharest sớm hơn nhưng không tối ưu tổng chi phí.
- Kết quả tìm được là `Arad → Sibiu → Fagaras → Bucharest → Urziceni → Hirsova`, với tổng chi phí là 633. Thuật toán "tham lam" này hướng đến đích nhanh hơn nhưng không đảm bảo tối ưu.

**Đối với thuật toán A*:**

- A* sử dụng đồng thời cả hai thành phần $f(n) = g(n) + h(n)$, trong đó $g(n)$ phản ánh chi phí thật đã đi và $h(n)$ là ước lượng còn lại đến đích.

- Khi đến Sibiu, thay vì chọn Fagaras như GBFS, A* chọn đi qua Rimnicu Vilcea rồi Pitesti, vì tổng $f(n)$ của nhánh này nhỏ hơn, đảm bảo tổng chi phí tối ưu.
- Đường đi tối ưu là `Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest → Urziceni → Hirsova`, với chi phí 601, đúng như kết quả chạy tay.




Hình 4: Đồ thị minh họa kết quả tìm đường: GBFS (xanh lá), A* (xanh dương)

---

## 4.3 Kết luận

- Kết quả chạy tay và chạy máy hoàn toàn trùng khớp, khẳng định việc cài đặt hai thuật toán là chính xác.
- **GBFS** có xu hướng "tham lam", chọn đường có heuristic nhỏ nhất tại mỗi bước, nên không luôn tối ưu.
- **A*** xem xét cả chi phí thực tế và ước lượng còn lại, nên tìm được đường đi tối ưu với tổng chi phí nhỏ hơn.

**Tổng kết:** Thuật toán A* là sự cải tiến vượt trội so với GBFS khi kết hợp giữa tính "tham lam" và "tối ưu toàn cục". Kết quả của cả hai phương pháp trên bản đồ Romania giúp minh họa rõ ràng sự khác biệt trong chiến lược tìm kiếm và hiệu quả của từng thuật toán.