TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

HOMEWORK 1

HỌC PHẦN: TRÍ TUỆ NHÂN TẠO NÂNG CAO

Giảng viên hướng dẫn : Ts. Đỗ Như Tài

Sinh viên thực hiện:

Phạm Văn Nam 3122410251

Thành phố Hồ Chí Minh - Tháng 10/2025

Mục lục

Question 1:

a)

Diễn tiến trạng thái (không đụng tường, rẽ khi v=0)

Trạng thái ban đầu: Start: (0,0, N, v=0)

1. right → (0,0, E, 0)
2. faster → v=1, đi 1 ô: (1,0, E, 1)
3. slow → v=0, đứng yên: (1,0, E, 0)
4. left → quay lên: (1,0, N, 0)
5. faster → (1,1, N, 1)
6. maintain → (1,2, N, 1)
7. maintain → (1,3, N, 1)
8. slow → v=0, đứng yên: (1,3, N, 0)
9. left → quay trái: (1,3, W, 0)
10. faster → (0,3, W, 1)
11. slow → v=0, đứng yên: (0,3, W, 0)
12. right → quay lên: (0,3, N, 0)
13. faster → (0,4, N, 1)
14. slow → v=0, đứng yên: (0,4, N, 0)
15. left → quay trái: (0,4, W, 0)

b)

$M \times N \times 4 \times (V\_max + 1)$
$\Rightarrow$ Size = $M \times N \times 4 \times (V\_max + 1)$.

c) Đáp án "NO"

Nếu theo giả định "mọi trạng thái đều reachable và hợp lệ"

Giải thích:

Nếu v = 0: luôn quay left hoặc right được (quay không di chuyển nên không va tường).

Nếu v > 0: luôn có thể slower về v−1; riêng khi v=1, slower cho v′=0 và di chuyển 0 ô — vẫn hợp lệ.
Nên mọi state đều có ≥1 action hợp lệ.

d) Nhánh tối đa (maximum branching factor) = 3.

- Ở v = 0: {left, right, faster} nếu ô trước mặt trống (maintain bị cấm khi v=0, slower cũng cấm vì v =0)
- Ở v > 0 và không vướng tường, v < V_max: {slower, maintain, faster}.
  Ví dụ state (x, y, E, v=1) ở vùng trống: actions = {slower, maintain, faster}.

e) Manhattan từ vị trí đến ô goal có admissible không? Không.
Manhattan không admissible trong bài toán này.

Giải thích: mỗi action có thể di chuyển nhiều ô (bằng vận tốc mới v'), nên số action tối thiểu (chi phí thật) có thể nhỏ hơn khoảng cách Manhattan (đếm ô). Vì vậy Manhattan có thể overestimate chi phí thật.

(f) Heuristic Manhattan / V_max admissible không? Có.
Vì mỗi action di chuyển tối đa V_max ô, nên để phủ quãng đường D phải cần ≥ D / V_max action (chưa kể tăng/giảm tốc, quay đầu chỉ làm tốn thêm). Do đó h = D / V_max ≤ chi phí tối ưu ⇒ admissible.

g) A* Tree search với heuristic inadmissible có ảnh hưởng tính đầy đủ (completeness) không (đồ thị hữu hạn, h ≥ 0)? Không.

(h) A* Tree search với heuristic inadmissible có thể làm mất tối ưu (optimality) không? Có.

i) Chọn lý do hợp lý để dùng inadmissible:

đáp án đúng:

- An inadmissible heuristic may be easier to compute, leading to a faster state heuristic computation time.
- An inadmissible heuristic can be a closer estimate to the actual cost (even if it’s an overestimate) than an admissible heuristic, thus exploring fewer nodes.

Question 2:

(a) BFS (tree search):

Tất cả (n) sao cho: d(n) < d(n_s).

(b) Uniform-Cost Search (tree search):

Tất cả (n) sao cho: g(n) < g(n_c).

(c) A* (giả sử (h) consistent):

Gọi (f(n)=g(n)+h(n)). Tất cả (n) sao cho: f(n) < f(n_c) = g(n_c).

Question 3:

a)

Algorithm	**A-C-E-G**	**A-C-E-F-G**	**A-B-D-E-F-G**	**Other**
UCS	☐(i)	☒ (ii)	☐ (iii)	☐ (iv)
Greedy with heuristic h1	☐(v)	☐ (vi)	☒ (vii)	☐ (viii)
Greedy with heuristic h2	☒ (ix)	☐ (x)	☐ (xi)	☐ (xii)
A* with heuristic h1	☐ (xiii)	☐ (xiv)	☒ (xv)	☐ (xvi)
A* with heuristic h2	☐ (xvii)	☒ (xviii)	(xix)	☐ (xx)

b) *Chi phí đường tối ưu (UCS) từ A đến G:*
11 (đường A–C–E–F–G).

c) *h1 admissible? consistent?*

Admissible ☐Yes ☒No

Consistent ☐Yes ☒No

- Admissible: Không (ví dụ C→G tối ưu =7 nhưng h1(C)=9).
- Consistent: Không (h1(C)=9 > c(C,E)=2 + h1(E)=3 ⇒ vi phạm).

d) *h2 admissible? consistent?*

Admissible ☒Yes ☐No

Consistent ☐Yes ☒No

- Admissible: Có (mọi h2 ≤ chi phí ngắn nhất đến G).
- Consistent: Không (h2(E)=5 > c(E,F)=3 + h2(F)=1 =4).

Question 4:

Phần a) [18 pts]

1. Depth-first tree-search on a finite graph is guaranteed to be complete.
Chọn đáp án: False

Giải thích: Bởi vì Depth-first tree search (DFS) không sử dụng tập đã duyệt (closed set) vì vậy trên đồ thị hữu hạn có chu kỳ, nó sẽ rơi vào vòng lặp vô hạn và không tìm thấy được mục tiêu. Ngay cả khi đồ thị hữu hạn, DFS cũng không đảm bảo hoàn thành nếu có chu kỳ.

2. Breadth-first tree-search on a finite graph is guaranteed to be complete.

Chọn đáp án: True

Giải thích: Do Breadth-first search (BFS) duyệt đồ thị theo từng mức (level-by-level). Trên đồ thị hữu hạn, BFS luôn tìm thấy mục tiêu nếu mục tiêu này tồn tại. Vì nó kiểm tra tất cả các nút ở độ sâu d trước khi chuyển sang độ sâu $d + 1$.

3. Iterative deepening tree-search on a finite graph is guaranteed to be complete.

Chọn đáp án: True

Giải thích: Iterative deepening kết hợp ưu điểm của BFS và DFS. Nó thực hiện DFS lặp lại với giới hạn độ sâu tăng dần và trên đồ thị hữu hạn, nó cũng đảm bảo hoàn thành vì cuối cùng sẽ đạt đến độ sâu chứa mục tiêu.

4. For all graphs without cycles, graph-search contains a larger frontier than tree-search.

Chọn đáp án: True

Giải thích: Bởi vì Graph-search sử dụng tập đã duyệt để tránh trùng lặp, nên frontier (biên) của nó chỉ chứa các nút chưa duyệt. Còn Tree-search không có tập đã duyệt, nên frontier có thể chứa nhiều nút trùng lặp, kiến cho kích thước frontier lớn hơn. Điều này đúng cho đồ thị không chu kỳ vì tree-search vẫn có thể tạo ra các nút trùng lặp.

5. Iterative deepening graph-search has the time complexity of BFS and the space complexity of DFS.

Chọn đáp án: True

Giải thích: Iterative deepening có độ phức tạp thời gian là $O(b^d)$ tương tự như BFS, với b là hệ số nhánh và d là độ sâu mục tiêu. Độ phức tạp không gian là $O(bd)$ thì giống với DFS vì nó lưu trữ đường đi từ gốc đến nút hiện tại.

6. If $h_1(s)$ is a consistent heuristic and $h_2(s)$ is a consistent heuristic, then $\min(h_1(s), h_2(s))$ must be consistent.

Chọn đáp án: False (Sai)

Giải thích:

Heuristic nhất quán phải thỏa mãn $h(n) \leq c(n, n') + h(n')$ cho mọi cạnh từ n đến n'. Ta xét ví dụ sau:

- Tại nút n, $h_1(n) = 5$, $h_2(n) = 3 \rightarrow h_{\min}(n) = 3$.
- Tại nút n', $h_1(n') = 2$, $h_2(n') = 4 \rightarrow h_{\min}(n') = 2$.
- Chi phí $c(n, n') = 1$.
- Kiểm tra: $h_{\min}(n) = 3 \leq c(n, n') + h_{\min}(n') = 1 + 2 = 3 \rightarrow$ thỏa mãn.
- Tuy nhiên, nếu tại n', $h_{\min}(n') = 1$ và $c(n, n') = 1$, thì $3 \leq 1 + 1 = 2$ là sai.

Do đó, $\min(h_1, h_2)$ không nhất thiết nhất quán.

Phần b) [5 pts]

Để heuristic là admissible nhưng không consistent, các giá trị heuristic phải thỏa mãn điều kiện không overestimate chi phí đến goal (admissible) và vi phạm ít nhất một điều kiện consistency (triangle inequality). Dựa trên đồ thị, chi phí thực từ các trạng thái đến goal G là:

- Từ S đến G: 4
- Từ A đến G: 3
- Từ C đến G: 2
- Từ G đến G: 0

Do đó để admissible thì:

h(S) ≤ 4

h(A) ≤ 3

h(C) ≤ 2

h(G) = 0.

Để vi phạm consistency, chọn h(S) = 4, h(A) = 0, h(C) = 0. Khi đó:

- Cạnh S-A: h(S) = 4 > cost(S,A) + h(A) = 1 + 0 = 1 → vi phạm consistency.
- Cạnh S-C: h(S) = 4 > cost(S,C) + h(C) = 2 + 0 = 2 → vi phạm consistency.

Các giá trị này vẫn đảm bảo admissibility.

Bảng điền giá trị heuristic:

State	$h(s)$
S	4
A	0
C	0
G	0