Homework Week 1

21280099 - Nguyễn Công Hoài Nam

Ngày 28 tháng 10 năm 2023

Đặt

b: Hệ số nhánh tối đa (số cây con tối đa của node).

d: Độ sâu của giải pháp đầu tiên mà thuật toán tìm được.

m: Độ sâu tối đa của cây.

BT 1: Breadth-first Search

Số Node mở rộng ở mỗi cấp độ

- 1 node ở cấp độ 0,
- b nodes ở cấp độ 1,
- b^2 nodes ở cấp độ 2,
- b^3 nodes ở cấp độ 3,
- ...
- b^d nodes ở cấp độ d.

Tìm kiếm dừng lại khi đạt đến cấp độ d, tổng số nodes bạn đã duyệt qua là:

$$1 + b + b^2 + \cdots + b^d = \frac{b^{d+1} - 1}{b - 1}$$

Vậy, Time Complexity = $O(b^{d+1})$ Số Node mở rộng chính là số Node cần lưu trữ nên Space Complexity cũng bằng Time Complexity

Complete	Với điều kiện b xác định
Time Complexity	$O\left(b^{d+1}\right)$
Space Complexity	$O\left(b^{d+1}\right)$
Optimal	Với đồ thị không có trọng số (tất cả trọng số đều bằng 1)

BT 2: Uniform-cost Search

Tìm kiếm dừng lại khi đạt đến cấp độ d, tổng số nodes bạn đã duyệt qua là:

$$1 + b + b^2 + \cdots + b^d = \frac{b^{d+1} - 1}{b - 1}$$

Nếu C^* là chi phí đến điểm đích và mỗi bước di chuyển giúp bạn tiến gần điểm đích ε đơn vị, số bước bạn cần thực hiện là $\frac{C^*}{\varepsilon} + 1$. Vì vậy, Time Complexity = $O(b^{(1+\frac{C*}{\varepsilon})})$. Số Node cần mở rộng chính là số Node cần lưu trữ nên Space Complexity = Time Complexity

Complete	Với b xác định và step-costs $\geq \epsilon$ với ϵ là một số dương > 0
Time Complexity	$O(b^{1+[C^*/e]})$
Space Complexity	$O(b^{1+[C^*/e]})$
Optimal	Có, luôn tìm đường đi có trọng số thấp nhất

BT 3: Depth-first Search

Tìm kiếm dừng lại khi đạt đến cấp độ m, tức là tất cả các Node trên cây.

$$1 + b + b^2 + \cdots + b^m = \frac{b^{m+1} - 1}{b - 1}$$

Vậy, Time Complexity = $O(b^m)$
BFS chỉ cần lưu trữ đường đi từ Node gốc đến Node lá nên số Node tối đa cần lưu trữ là bm
Vậy nên, Space Complexity = $O(bm)$

Complete	Có, trên đồ thị hữu hạn. Không, nếu có một đường đi vô hạn mà không có giải pháp.
Time Complexity	$O(b^m)$
Space Complexity	$O(bm)$
Optimal	Không (Có thể mở rộng một đường đi dài hơn nhiều so với đường đi tối ưu đầu tiên).

BT 4: Greedy Best-first Search

Time Complexity: Trong mỗi bước, cần xem xét tối đa b trạng thái con, với độ sâu tối đa là m. Do đó, phức tạp thời gian là $O(b^m)$.
Space Complexity: Trong trường hợp xấu nhất, cần lưu trữ tất cả Node con từ gốc đến đích là ($O(b^m)$.
Complete: Không đảm bảo tìm ra giải pháp nếu nó tồn tại trong không gian tìm kiếm. Thuật toán chỉ dựa vào hàm heuristic để chọn nút tiếp theo, không xem xét chi phí đã đi qua, có thể dẫn đến các vòng lặp không kết thúc.
Optimal: Không đảm bảo tìm ra giải pháp có chi phí thấp nhất. Hàm heuristic chỉ ước lượng từ trạng thái hiện tại đến mục tiêu, không cung cấp thông tin về chi phí đã đi qua. Nếu có đường dẫn chi phí thấp nhất nhưng không được chọn, thuật toán có thể trả về giải pháp không tối ưu.

Complete	Không
Time Complexity	$O(b^m)$
Space Complexity	$O(b^m)$
Optimal	Không

BT 5: Proof Optimality of A^*

Theorem: Nếu $h(n)$ là một heuristic chấp nhận được (admissible), việc sử dụng thuật toán A* với TREE-SEARCH là tối ưu.
Proof:

$$\begin{aligned} f(G_2) &= g(G_2) \text{ vì } h(G_2) = 0 \\ g(G_2) &> g(G) \text{ vì } G_2 \text{ là suboptimal} \\ f(G) &= g(G) \text{ vì } h(G) = 0 \\ f(G_2) &> f(G) \text{ từ các điều trên} \\ h(n) &\leq h^*(n) \text{ vì } h \text{ là admissible} \\ g(n) + h(n) &\leq g(n) + h^*(n) \\ f(n) &\leq f(G) \end{aligned}$$

Đặt

$$f(G) = C^*$$

Ta được

$$f(G2) > C^* \qquad f(n) < C^*$$

Do đó A^* sẽ không chọn G_2 để mở rộng và A^* luôn là một lời giải tối ưu