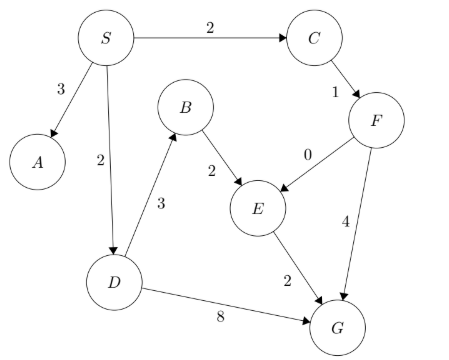
Nguyễn Thiên Thuận  
MSSV: 3122410403

1 Search Algorithms (20 pts)

**

Using each of the following graph search algorithms presented in lecture, write out the order in which nodes are added to the explored set, with start state *S* and goal state *G*. Break ties in alphabetical order by *the last state in the path*. Additionally, what is the path returned by each algorithm? What is the total cost of each path?

(a) Breadth-first

Explored order: S, A, C, D, F, B, G

Path found: S → D → G

Total cost: 2 (S→D) + 8 (D→G) = 10

(b) Depth-first

Explored order: S, A, C, F, E, G

Path found: S → C → F → E → G

Total cost: S→C (2) + C→F (1) + F→E (0) + E→G (2) = 5

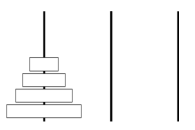
(c) Iterative deepening

Explored order: S, A, C, F, D, B, G

Path found: S → D → G

Total cost: S→D (2) + D→G (8) = 10

2 Tower of Hanoi (20 pts)



The Tower of Hanoi is a canonical puzzle studying problem solving and formulation. The puzzle starts with *n* disks of different sizes stacked in order of size (see picture above) on a peg, along with two empty pegs. We can move disks freely between the pegs, but larger disks cannot be stacked on top of smaller ones. The goal is to move all disks to the third peg.

We will attempt to formulate Tower of Hanoi as a search problem.

(a) How could we represent this puzzle as a problem, ie. what would the states be?

Một state (trạng thái) trong bài toán Tháp Hà Nội có thể được biểu diễn bằng vị trí của mỗi đĩa - tức là, mỗi đĩa nằm ở cọc nào (A, B, hay C).

Cụ thể:

* Nếu ta có n đĩa, mỗi đĩa có thể ở 1 trong 3 cọc.
* Vậy một trạng thái có thể biểu diễn bằng bộ n phần tử (tuple), trong đó phần tử thứ i cho biết đĩa thứ i đang ở cọc nào.

Ví dụ:

Ta đặc cọc là A, B, C.

* Trạng thái khởi đầu (initial state): tất cả các đĩa nằm ở cọc A → (A, A, A)
* Trạng thái đích (goal state): Tất cả các đĩa nằm ở cọc C → (C, C, C)
* Một trạng thái trung gian có thể là: (C, A, A) nghĩa là:
  + Đĩa nhỏ nhất ở cọc C
  + Hai đĩa lớn hơn vẫn ở cọc A

For the following questions, assume that you have used your state representation in your answer above. (b) What is the size of the state space in terms of *n*?

Từ phần a, ta đã biểu diễn một trạng thái bằng cách cho biết mỗi đĩa đang nằm ở cọc nào (A, B hoặc C)

* Mỗi đĩa có 3 lựa chọn: A, B hoặc C
* Có n đĩa → tổng số cách sắp xếp = 3^n

Do đó, kích thước không gian trạng thái là 3^n

(c) What is the starting state?

Tất cả các đĩa đề ở cọc A → start = (A, A,..., A)

(d) From some given state, what legal actions are there?

Trong bài toán Tháp Hà Nội, một hành động hợp lệ là di chuyển (move) một đĩa từ cọc này sang cọc khác, miễn là tuân thủ quy tắc sau:

“Không được đặt đĩa lớn lên đĩa nhỏ”

Giả sử trạng thái hiện tại:

A: [3, 2]

B: [ ]

C: [1]

(Thứ tự trong ngoặc vuông là từ đáy lên đỉnh)

→ Các hành động hợp lệ là:

* Di chuyển đĩa 1 từ C → B (hợp lệ, vì B trống)
* Di chuyển đĩa 1 từ C → A (không hợp lệ, vì đĩa 2 nhỏ hơn đĩa 1)
* Di chuyển đĩa 2 từ A → B (hợp lệ, vì B trống)

(e) What is the goal test? Remember that this determines whether a given state is a goal state, goalTest(state).

Trong bài toán Tháp Hà Nội, mục tiêu là:

Di chuyển tất cả các đĩa từ cọc ban đầu (A) sang cọc thứ ba (C),

vẫn tuân thủ quy tắc đĩa lớn không được nằm trên đĩa nhỏ hơn.

Nếu ta biểu diễn một trạng thái như sau:

State = (peg\_of\_disk1, peg\_of\_disk2,..., peg\_of\_diskN)

Thì goal state sẽ là: (C, C, C,.., C)

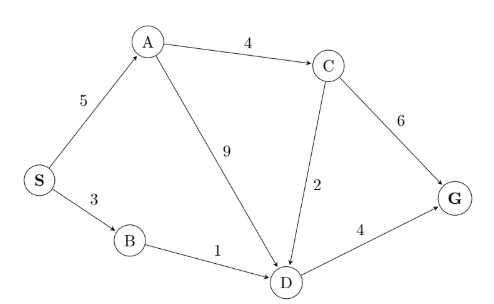
goalTest(state) trả về True khi mọi đĩa đều ở cọc C, tức là trạng thái = (C, C, C,..., C)

3 Designing & Understanding Heuristics (20 pts)

Today, we will be taking a closer look at how the performance of *A∗*is affected b y t he h euristics i t u s es. To do this, we’ll be using the graph below. You may have noticed that no heuristic values have been provided (*Recall:* What is *A∗* without heuristic values?). This is because we’ll be working together to come up with heuristics ourselves!

In groups, design both an admissible heuristic and a consistent heuristic for the graph below by anno tating each node with a heuristic value. (Note: you do NOT need to find a c losed f orm w ay t o r epresent the heuristic function.)

When you have completed your heuristics, work together to answer the questions below.



(a) Write down the path found by running *A∗* using your heuristic on the graph above.

Heuristic admissible nhưng không có consistent: S:7, A:9, B:5, C:5, D:2, G:0

Heuristic consistent = exact distances: S:8, A:10, B:5, C:6, D:4, G:0

Thứ tự các nút được mở rộng (explored order): S, B, D, G

(b) Work with your group to come up with a heuristic that’s admissible but not consistent.

Bộ giá trị heuristic cho từng nút như sau:

| h(S) | 7 |
| --- | --- |
| h(A) | 9 |
| h(B) | 5 |
| h(C) | 5 |
| h(D) | 2 |
| h(G) | 6 |

Admissible: Vì các giá trị này không vượt quá chi phí ngắn nhất thực tế từ nút tới G (những chi phí thực tế h\* ta biết là: S:8, A:10, B:5, C:6, D:4, G:0)

Không consistent: Để consistent cần thỏa mãn cho mọi cạnh u → v: h(u) <= cost(u, v) +h(u)

Xét cạnh C → D (cost = 2):

* h(C) = 5 nhưng cost(C,D) + h(D) = 2 + 2 = 4
* Ta có 5 <= 4 tức là bị vi phạm

Vậy heuristic trên là hợp lệ (admissible) nhưng không thỏa tính consistent.

(c) (Bonus) Explain why a consistent heuristic must also be admissible. You may assume that the heuristic value at a goal node is always 0.

Một heuristic consistent luôn admissible, vì:

* Với nút đích G, .
* Với các nút khác, do tính nhất quán:

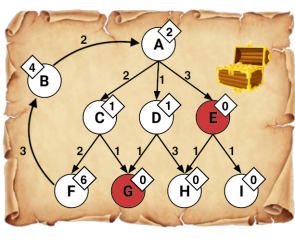


⇒ theo quy nạp dọc theo đường đi ngắn nhất tới G, tổng cộng ta luôn có  
 .  
 → nên consistent ⇒ admissible.

2

4 Treasure Hunting (20 pts)

We are lost at sea, trying to find a h i dden t r easure. W e h a ve a tr easure m a p t h at t e lls u s w h ich p a ths we can take and approximately how far away the treasure is but it’s not very accurate. We do know that the map will never overestimate the distance to the treasure. There is treasure on two different islands (b ut we only need to reach one of them).



*A* is the the island where we are currently and the shaded red states are the locations of the treasure. Arrows encode possible actions from each island, and numbers by the arrows represent action costs. Note that arrows are directed; for example, *A → C* is a valid action, but *C → A* is not. Numbers shown in diamonds are heuristic values that estimate the optimal (minimal) cost to get from that island to any treasure.

Run each of the following search algorithms with graph search and write down the nodes that are added to the explored set during the course of the search, as well as the final path returned and the correspond ing cost of the final path, if applicable. When popping off of the frontier, assume that ties are broken alphabetically.

(a) Depth-First Search

Explored set: A → B → F → C → G

Path returned: A → C → G

Total cost: 2 (A→C) + 1 (C→G) = 3

3

(b) Breadth-First Search

Explored set: A, B, C, D, E, F, G

Path returned: A → C → G

Total cost: 3

(c) Uniform-Cost Search

Explored set: : A, D, B, C, G

Path returned and cost: : A → D → G

Total cost: 1 (A→D) + 1 (D→G) = 2

(d) Greedy Search

Explored set: A, E

Path returned and cost: A → E

Total cost: A → E(3)

(e) A*∗* Search

Explored set: A, D, G

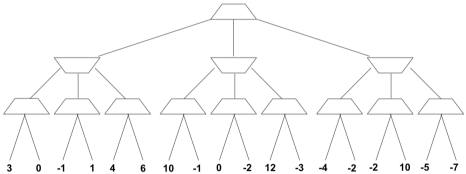
Path returned and cost: A → D → G

Total cost: 1 (A→D) + 1 (D→G) = 2

4

5 Adversarial Search (10pts)

Consider the following game tree, where the root node is a maximizer. Using alpha beta pruning and visiting successors from left to right, record the values of alpha and beta at each node. Furthermore, write the value being returned at each node inside the trapezoid. Put an ‘X’ through the edges that are pruned off.

5

6 True/False Section (10pts)

For each of the following questions, answer true or false and provide a brief explanation (or counterexample, if applicable).

(a) Depth-first s earch a lways e xpands a t l east a s m any n odes a s *A ∗*s earch w ith a n a dmissible heuristic.

(b) Assume that for a single move, a rook can move any number of squares on a chessboard in a straight line, either vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the smallest number of moves to move the rook from square A to square B.

(c) Euclidean distance is an admissible heuristic for Pacman path-planning problems. (d) The sum of several admissible heuristics is still an admissible heuristic.

(e) Admissibility of a heuristic for *A∗*search implies consistency as well.

(f) *A∗* with graph search is always optimal with an admissible heuristic.

For (g) and (h), consider an adversarial game tree where the root node is a maximizer, and the minimax value of the game (i.e., the value of the root node after running minimax search on the game tree) is V*M*. Now, also consider an otherwise identical tree where every minimizer node is replaced with a chance node (with an arbitrary but known probability distribution). The expectimax value of the modified game tree is V*E*.

(g) V*M* is guaranteed to be less than or equal to V*E*.

(h) Using the optimal minimax policy in the game corresponding to the modified (chance) game tree is guaranteed to result in a payoff of at least V*E*.

6