BÀI TOÁN BIẾN ĐỔI CHUỖI KÝ TỰ

Nguyễn Duy Khang, Vũ Thanh Lâm

Trường đại học Sư phạm Kỹ thuật TP.HCM

Tóm tắt

Trong thời đại 4.0 hiện nay, vấn đề bảo mật thông tin trở nên quan trọng hơn bao giờ hết. Vì thế kỹ thuật "mã hóa" thông tin đã xuất hiện. Mã hóa sẽ mang lại tính an toàn cao hơn cho thông tin khi mà thông tin phải đi qua nhiều trạm, nhiều sever khác nhau để tới được đích. Nếu không có mã hóa thông tin thì việc thông tin bị lộ dẫn đến những sự cố to lớn ngoài ý muốn có thể diễn ra. Bài toán như sau cũng là một cách mã hóa thông tin nhưng ở quy mô nhỏ hơn cụ thể là "Bắt đầu bằng một chuỗi các ký tự A,B,C, và E (ví dụ ABABAECCEC). Chuỗi này có thể được biến đổi bằng các luật AC=E, AB=BC, BB=E và Ex=x với mọi ký tự x. Viết chương trình từ ký tự bất kỳ, xuất ra một dãy các biến đổi để chỉ còn ký tự E. Ví dụ ABBC có thể biến đổi thành AEC --> AC -->E".

Người nhập sẽ đưa vào 1 chuỗi gồm các ký tự A,B,C và E, người viết chương trình sẽ sử dụng các thuật toán và các luật hurictis để biến đổi chuỗi còn lại duy nhất 1 ký tự là E, trường hợp không tìm ra giải pháp sẽ xuất "no solution".

Từ khóa: biến đổi, điểm.

1. Đặt vấn đề

Bài toán biến đổi chuỗi yêu cầu tìm ra đường đi đến đích. Vậy bài toán này phải sử những thuật toán tìm kiếm có lưu đường đi tới đích như: depth first search, breadth first search… và có thể so sánh các thuật toán xem thuật toán nào tối ưu nhất.

2. Giải pháp xử lý

2. 1. Dữ liệu

Dữ liệu đầu vào của bài toán là một hoặc nhiều chuỗi ngẫu nhiên hoặc tự nhập với đôj dài tùy ý thuộc tập hợp các ký tự {‘A’, ‘B’, ‘C’, ‘E’}. Với dữ liệu ngẫu nhiên với độ dài chuỗi càng dài càng khó tạo ra các chuỗi biến đổi ra yêu cầu. dữ liệu đầu ra là đường đi từ chuỗi ký tự đầu vào hoặc là bảng thời gian thực thi của thuật toán.

2. 2. Cấu trúc kiểu dữ liệu ký tự

```
typedef struct NodeString
{
        char letter[100];
        Actions action;
        int value;
        int depth;
        NodeString* parent;
        NodeString* nextNode;
} Node;
```

Giải thích: kiểu dữ liệu ký tự chứa những dữ liệu cần thiết gồm: letter để chứa chuỗi, action để chứa hành động để chuỗI cha biến đổi thành nó, value chứa chi phí hỗ chợ tìm kiến bằng thuật toán A*, depth để chứa chiều sâu, parent chứa địa chỉ cha của nó để xuất đường đi từ chuỗI đầu vào đến nó, nextNode để chứa địa chỉ tiếp theo để tạo tạo thành list.

2. 3. Các giả thuật tìm kiếm

Bài toán biến đổi chuỗi sử dụng các thuật toán tìm kiếm depth first search, breadth first search và thuật toán a* algorithm để tìm kiếm ra đường đi đến đích.

Bảng 1: Các thuật toán tìm kiếm

Giải thuật	Mã giả	Cách thức thực hiện

DFS	Procedure DFS(g,v) is
	let S be a stack
	S.push(v)
	while S is not empty do
	v = S.pop()
	if v is not labeled as discovered then
	label v as discovered
	for all edges from v to w in G.adjacentEdges(v) do
	S.push(w)
		Bắt đầu bằng việt khởi tạo stack S chứa các chuỗi ký tự, cho chuỗi đầu vào vô S.
		Sau đó tạo vòng lặp while với điều kiện dừng là S rỗng, trong vòng lặp lấy phần tử vào S sau cùng - v ra kiểm tra có phải chuỗi đích {'E'} hay không, nếu đúng thì trả v về, không thì kiểm tra các chuỗi con của v chuỗi nào chưa xét và chưa có trong S thì bỏ vào S.

BFS	Procedure BFS(g,v) is
	let Q be a queue
	Q.push(v)
	while S is not empty do
	v = Q.pop()
	if v is not labeled as discovered then
	label v as discovered
	for all edges from v to w in G.adjacentEdges(v) do
	Q.push(w)
		Bắt đầu bằng việt khởi tạo queue Q chứa các chuỗi ký tự, cho chuỗi đầu vào vô Q.
		Sau đó tạo vòng lặp while với điều kiện dừng là Q rỗng, trong vòng lặp lấy phần tử vào Q đầu tiên - v ra kiểm tra có phải chuỗi đích {'E'} hay không, nếu đúng thì trả v về, không thì kiểm tra các chuỗi con của v chuỗi nào chưa xét và chưa có trong Q thì bỏ vào Q.

A*	Procedure A*(g,v) is let L be a list L.add(v)	Bắt đầu bằng việt khởi tạo list L chứa các chuỗi ký tự, cho chuỗi đầu vào vô L.

	while S is not empty do
	v = L.minCost()
	if v is not labeled as discovered then
	label v as discovered
	for all edges from v to w in G.adjacentEdges(v) do
	L.push(w)
		Sau đó tạo vòng lặp while với điều kiện dừng là L rỗng, trong vòng lặp lấy phần tử có chi phí (với chi phí tính bằng các hàm heuristic) nhỏ nhất trong L - v ra kiểm tra có phải chuỗi đích {'E'} hay không, nếu đúng thì trả v về, không thì kiểm tra các chuỗi con của v chuỗi nào chưa xét và chưa có trong L thì bỏ vào L.

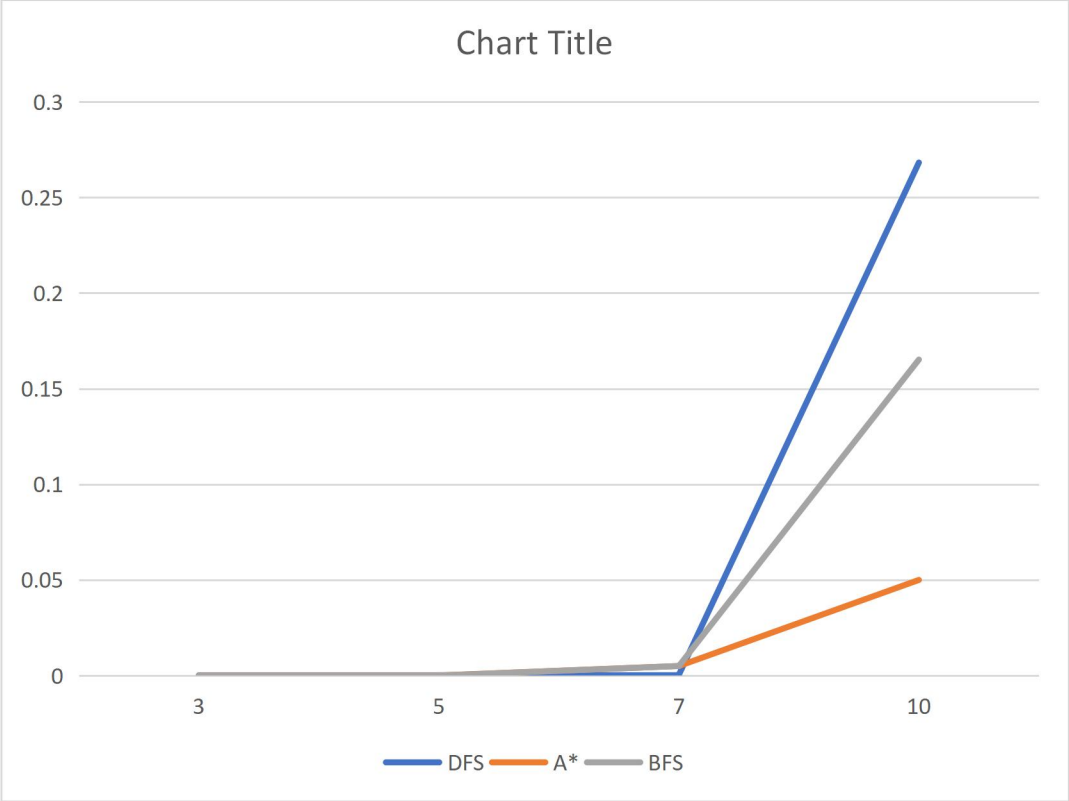
2. 4. Các hàm heuristic

Các hàm heuristic sẽ trả về số điểm của chuỗi, chuỗi có giá trị càng nhỏ thì có độ ưu tiên càng lớn.

Bảng 2: các hàm heuristic

TT	Tác dụng	Mô tả
1	Ưu tiên hành động	Cho hành động Ex điểm là 0, hành động AC điểm là 1, hành động BB điểm là 2, hành động AB điểm là 3
2	Ưu tiên chuỗi chứa nhiều phần tử biến đổi được thành E	Điểm bằng độ dài chuỗi trừ đi số phần tử biến đổi được
3	Ưu tiên chuỗi ngắn	Điểm bằng độ dài chuỗi

3. III. Kết quả



Bảng 3: thời gian thực thi trung bình 100 chuỗi các thuật toán

Độ dài / Thuật toán	3	5	7	10
DFS	0.0001700	0.0002000	0.0004100	0.2682000
BFS	0.0000000	0.0000000	0.0050140	0.1652000
A*	0.0000000	0.0000000	0.0050000	0.0500000

Nhận xét: Sau quá trình thực hiện so sánh, đánh giá thời gian thực hiện của các giải thuật để rút gọn chuỗi, nhóm chúng em đưa ra nhận xét như sau: Thời gian thực hiện của các giải thuật BFS > DFS > A.

4. IV. Kết luận

Ưu điểm: đã giải quyết được vấn đề bài toán đặt ra bằng cách áp dụng các thuật toán đã học.

Nhược điểm: vẫn còn giới hạn số lượng ký tự đầu vào, có trường hợp thực hiện khá lâu dẫn đến treo máy.

Hướng phát triển: Mở rộng bài toán với số lượng ký tự đầu vào lớn, tiếp tục nâng cấp để áp dụng mã hóa những vấn đề phức tạp hơn trong tương lai.

5. VI. Tài liệu tham khảo

1. https:// viblo.asia/p/cac-thuat-toan-co-ban-trong-ai-phan-biet-best-first-search-va-uniform-cost-search-ucs-Eb85omLWZ2G [2]

2. https://vi.wikipedia.org/wiki/Giải_thuật_tìm_kiếm_A*#Ý_tưởng_trực_quan[3]

3. https://vi.wikipedia.org/wiki/Tìm_kiếm_theo_chiều_sâu[1]