BÀI TẬP LẬP TRÌNH GIỮA KỲ MÔN TOÁN RỜI RẠC

Mã sinh viên: 202416913

Họ và tên: Phan Đức Hiếu

Lớp: Việt – Nhật 04 K69

Bài tập 1

1. Cách biên dịch:

INPUT:

- Không nhập gì cả

OUTPUT:

- Gồm nhiều dòng mỗi dòng là một trạng thái kết thúc (theo yêu cầu để bài) mà qua một dãy các thao tác từ (0, 7, 4) có thể đến được.

2. Một số test thành công:

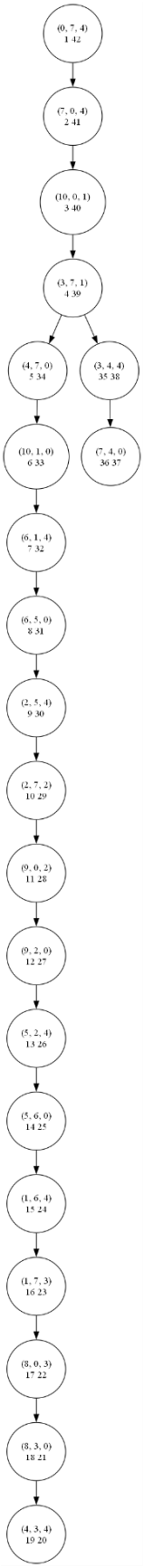
INPUT	OUTPUT
	2 7 2 9 0 2 9 2 0 5 2 4

3. Thông tin bổ sung *(hình vẽ ở trang dưới)*

Sau đây là đoạn code dùng để tạo các đỉnh, cạnh trong graphviz:

https://ideone.com/BQON0o

Hình bên là cây DFS bắt đầu từ đỉnh (0, 7, 4) bên dưới mỗi đỉnh là trạng thái start và post phân cách bởi khoảng trắng.



Bài tập 2

1. Cách biên dịch:

INPUT:

- Từ dòng 1 đến dòng 5757 là các ký tự trong tập sau, mỗi từ trên một dòng https://www-cs-faculty.stanford.edu/~knuth/sgb-words.txt
- Dòng thứ 5758 là hai từ cần tìm đường đi ngắn nhất.

OUTPUT:

- Dòng thứ 1: là số thành phần liên thông.
- Dòng thứ 2: đường đi ngắn nhất (nếu có) giữa hai từ đã nhập. Nếu trong hai từ có từ không có trong danh sách thì in ra -1, không tồn tại đường đi ngắn nhất thì ghi ra -2.
- Dòng thứ 3: một đường đi ngắn nhất thỏa mãn (nếu có).

2. Một số test thành công:

INPUT	OUTPUT
which … pupal words graph	853 8 words lords loads goads grads grade grape graph

INPUT	OUTPUT
which … pupal pupal which	853 13 pupal papal papas paras parts warts waits whits whips whipt whist whish which

INPUT	OUTPUT
which … pupal pupal whic	853 -1

INPUT	OUTPUT
which … pupal pupal osier	853 -2

3. Thông tin bổ sung:

a. CTDL cây Trie:

Giới thiệu

Trie, hay một số tài liệu còn gọi là cây tiền tố, là một cấu trúc dữ liệu dạng cây hữu dụng được dùng để quản lí một tập hợp các xâu. Mặc dù dễ hiểu và dễ cài đặt, trie lại có rất nhiều ứng dụng. Do vậy, trie thường xuyên xuất hiện trong các cuộc thi lập trình ở Việt Nam nói riêng và quốc tế nói chung.

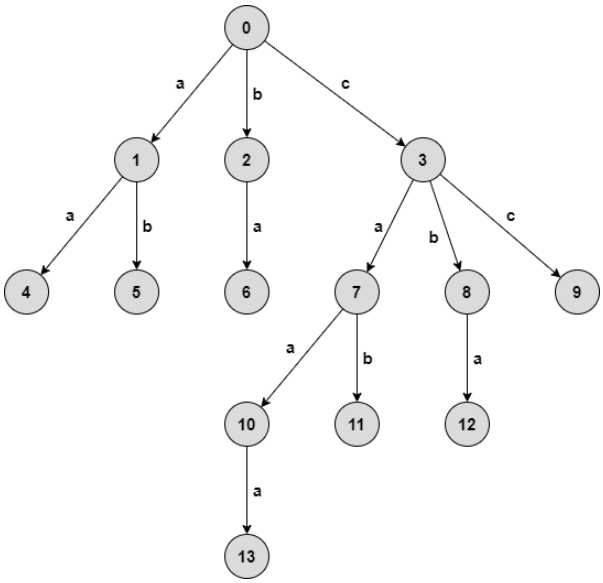
Một trie cơ bản có thể thực hiện ba thao tác sau với độ phức tạp thời gian tuyến tính:

- Thêm một xâu vào tập hợp
- Xóa một xâu khỏi tập hợp
- Kiểm tra một xâu có nằm trong tập hợp đó hay không

Cấu trúc

Trie là một cấu trúc dữ liệu dạng cây dùng để lưu trữ một danh sách các xâu với bộ kí tự hữu hạn, cho phép việc lưu trữ các xâu hiệu quả có tiền tố giống nhau.

Hãy xem xét một ví dụ sau:



Trong một trie, mỗi cạnh được biểu diễn bằng một kí tự, mỗi đỉnh và đường đi từ gốc đến đỉnh đó biểu diễn một xâu gồm các kí tự thuộc các cạnh trên đường đi đó. Ví dụ, đỉnh biểu diễn xâu ab, đỉnh biểu diễn xâu caa.

Cấu trúc của trie rất dễ hiểu và cài đặt. Gọi child(u, c) là đỉnh con của đỉnh được nối bởi cạnh được biểu diễn bằng kí tự , hoặc bằng nếu đỉnh con đó không tồn tại. Xâu được thể hiện bởi đỉnh con này sẽ chính là xâu được thể hiện bởi đỉnh , thêm kí tự vào cuối. Do vậy, ta chỉ cần mảng child này với mỗi đỉnh để duy trì cấu trúc của trie. Ví dụ, trong ảnh trên, child(1, 'b') = 5, child(3, 'c') = 9, child(11, 'b') = -1.

b. Kiểm tra sự đầy đủ của việc xây dựng đồ thị:

Chương trình sau có nhiệm vụ kiểm tra xem danh sách kề đã xây dựng như trong chương trình Ex2.cpp đã đủ so với yêu cầu của đề bài hay chưa. Và theo như kết quả trả ra thì đồ thị đã được xây dựng đầy đủ.

https://ideone.com/h8hr6l

INPUT:

- các từ trong tập.

OUTPUT:

- Nếu đồ thị đủ in *"Da nap du do thi".*
- Ngược lại in *"Chua nap du do thi"*.

Bài tập 3

1. Cách biên dịch:

INPUT:

- Từ dòng 1 đến dòng 5757 là các ký tự trong tập sau, mỗi từ trên một dòng https://www-cs-faculty.stanford.edu/~knuth/sgb-words.txt
- Dòng thứ 5758 chứa 1 từ (theo yêu cầu 2)
- Dòng thứ 5759 chứa 2 từ u, v lần lượt là từ bắt đầu và kết thúc (theo yêu cầu 3)

OUTPUT:

- Dòng thứ 1: là số thành phần liên thông mạnh.
- Dòng thứ 2: Tất cả các đỉnh thuộc cùng thành phần liên thông mạnh với từ đã nhập hoặc -1 nếu số đã nhập không thuộc tập.
- Dòng thứ 3: một đường đi ngắn nhất thỏa mãn (nếu có). In -1 nếu một trong hai từ u, v không có trong tập. In -2 nếu không tồn tại đường đi ngắn nhất.

2. Một số test thành công:

INPUT	OUTPUT
which … pupal giddy diddy giddy	763 giddy middy biddy diddy diddy giddy

INPUT	OUTPUT
which … pupal giddy diddy biffy	763 giddy middy biddy diddy -2

INPUT	OUTPUT
which … pupal giddy diddy boffo	763 giddy middy biddy diddy -1

INPUT	OUTPUT
which … pupal gloss gloss rumba	763 gloss floss slobs blobs blocs clogs flogs golfs globs slogs gloss lasso soars orcas scram charm harum rumba

INPUT	OUTPUT
which … pupal glosh gloss rumba	763 -1 gloss lasso soars orcas scram charm harum rumba

3. Thông tin bổ sung:

a. Tư tưởng thuật toán trong xây dựng đồ thị

Đề bài yêu cầu xây dựng đồ thị G mà đỉnh u có cạnh nối đến đỉnh v, nếu 4 ký tự cuối của từ u xuất hiện đầy đủ trong từ v:

Cách 1: duyệt từng cặp từ phân biệt và kiểm tra, cách này mất O(5757 * 5757 * (5 + 5 + 26)) trong đó O(5 + 5 + 26) là độ phức tạp để kiểm tra sự phù hợp của hai từ.

Cách 2: Với mỗi một từ u trong tập, chỉ xét tập S gồm 4 ký tự cuối cùng, thực hiện hoán vị tập S rồi thêm các ký tự 'a' đến 'z' vào từng vị trí trong các hoán vị để tạo thành tập X là tập các từ mà u có thể tồn tại cạnh nối. Sau đó phải kiểm tra xem các từ trong X có nằm trong tập để thành lập cạnh. Cách này tốn (5757 * 4! * 5 * 26 * 5) với O(5) ở cuối đã là tối ưu khi sử dụng CTDL Trie để tìm trong tập. Kết luận: Độ phức tạp vẫn tương đối lớn

Cách 3 (cách dùng trong code): Do đề bài chỉ đề cập đến sự "xuất hiện" nghĩa là không có tính thứ tự, do vậy có thể sắp xếp các từ để giảm bớt việc phải sinh các hoán vị như trên.

- Bước 1: Tiền xử lý. Với mỗi từ trong tập, lần lượt bỏ từng ký tự và sắp xếp các từ còn lại sau đó add vào cây Trie. Độ phức tạp O(5757 * 5 * 26 * 5), với O(26) là sắp xếp, O(5) là tìm kiếm trong Trie.
- Bước 2: Với mỗi từ u trong tập, chỉ lấy 4 ký tự cuối và sắp xếp chúng. Tìm trong cây Trie, nếu các từ trong Trie mà trùng với từ đã sắp xếp thì tập các từ được lưu trong danh sách "next" là tập các từ có 4 ký tự trùng với 4 ký tự cuối của từ u. Độ phức tạp O(5757 * 26 * 5) với O(26) là sắp xếp, O(5) là tìm kiếm trong Trie.
- Kết luận: Tổng độ phức tạp của thuật toán xây dựng đồ thị trong cách 3 nhanh hơn gấp 10 lần so với cách 2 và gấp 1000 lần so với cách 1.

b. Kiểm tra thuật toán xây dựng đồ thị

Chương trình sau có nhiệm vụ kiểm tra xem danh sách kề đã xây dựng như trong chương trình Ex3.cpp đã đủ so với yêu cầu của đề bài hay chưa sử dụng Cách 1 để kiểm tra Cách 3. Và kết quả trả ra đã chứng minh thuật toán xây dựng đồ thị như trong Ex3.cpp là đầy đủ với yêu cầu đề bài.

https://ideone.com/MW1tpJ

INPUT:

- các từ trong tập.

OUTPUT:

- Nếu đồ thị đủ in *"Da nap du do thi"*.
- Ngược lại in *"Chua nap du do thi"*.