UNOFFICIAL QUIZ *for* PRACTICE SOLUTIONS

QUIZ 11

Computer Science 61A . November 19, 2015 . alvinwan.com/cs61a

This quiz will not count towards your grade. It exists to simply gauge your understanding. You will have 5 minutes to complete this quiz. In that timespan, your goal is to complete the first question and at least attempt the second.

01. CALLING ALL SQUARES

Write a SQL statement that finds all perfect squares from 156 to 1145.

```
> select * from squares  # expected output

169
196
...
1024
1089

create table squares as  # code here

  with

    i(n) as (

      select 1 union

      select n+1 from i where n < 1145

    )

  select b.n as n

    from i as a, i as b

    where a.n*a.n=b.n and b.n > 156;
```

02. SUMUKH PHONE HOME

Sumukh is trying to escape an angry turkey horde and a Thanksgiving gone horribly wrong. He has a map G containing all nodes between him and the escape spaceship. However:

1. All nodes only allow travel in one direction, meaning he can walk from a to b but not b to a. You can assume that no cycles exist.
2. Sumukh can only move 2x as fast as a turkey. He can only travel at maximum 300 mi. before the swarm of turkeys close in on him.

Write a SQL statement that, given a table G of nodes, will generate all possible paths to the spaceship, along with the path's total distance. All paths should end at "spaceship".

```
> .schema
G(str start, str finish, int dist)  # added datatypes

> select * from paths # expected output

Soda dungeon,San Francisco,spaceship|100
Cory,Prof. DeNero’s office,San Jose,San Francisco,spaceship|150
...

create table paths as  # explanation on next page

  with

    candidates(start, finish, path, dist) as (

      select start, finish, start|","|finish, dist from G union

      select p1.start, p2.finish, p1.path|","|p2.path, p1.dist + p2.dist

        from candidates as p1, G as p2

        where p1.finish = p2.start and p1.finish != "spaceship"

          and p1.dist + p2.dist < 300

    )

  select path, dist as distance from candidates where finish = “spaceship”;
```

Explanation: Let us tackle this problem in a systematic fashion. First, we'll discuss the base case and then the recursive case. For each case, we'll (1) determine the data we want, (2) determine what subset of that data we want, and (3) write the "body" of function call.

Base case

What is the simplest, shortest path possible? Why, using only one path from G of course. Let's consider each of the three steps outlined above:

1. What data do we want? G - it contains all of the basic paths we can take.

```
select ?? from G where ??
```

2. What subset of G do we want? All of it! Since we're just copying data over, we want all paths. (Side note: we could add a `where` here to enforce `dist < 300`, but let's assume that no path on its own is longer than 300. If this note is confusing, just ignore it.)

`select ?? from G` ~~**`where ??`**~~

3. What is the body of the function call? Again, since we're just copying data over, the base case just formats the data according to the way we want it. We want `start, finish, path, dist`. So let's copy **`start, finish`**. Then, create the new `path` string by concatenating the `start` and `finish` together with a comma in between - **`start | "," | finish`**. Finally, copy **`dist`**.

```
select start, finish, start|","|finish, dist from G
```

Recursive case

In each recursive call, we'd like to elongate these paths until we (a) reach the spaceship or (b) have travelled too far, but how can we *do that*? In sum, we will consider a path we've already built and add just one more short path from G!

1. What data do we want? candidates - the paths we're building, and G - all of the basic paths we can consider adding to the paths we're building. We're going to now name candidates as p1 and G as p2.

```
select ?? from candidates as p1, G as p2 where ??
```

2. What subset of candidates and G do we want? We have three restrictions. First, we don't want to extend a path that has already reached the spaceship, so **`finish != "spaceship"`**. Second, we don't want to travel farther than Sumukh possibly can, so **`p1.dist + p2.dist < 300`**. Third, we have to make sure that the path from G we're considering is legitimate. If we've ended up at Cory, for example, we only want paths that start from Cory. Since, candidates is the path we're building and G is the short path we're considering taking, the end of candidates must match with the start of G. Remembering that candidates is p1 and G is p2, **`p1.finish = p2.start.`**

```
select ?? from candidates as p1, G as p2 where p1.finish = p2.start and
p1.dist + p2.dist < 300 and finish != “spaceship”
```

3. What is the body of the function call? We’d like to continue building our paths in candidates using the possible short paths from G. The path we’re building doesn’t change its `start`. So, we use **`p1.start`** for our start. However, we’re travelling to a new destination, per the path we consider from G. So, our new finish is **`p2.finish`**. Just like we did in the base case, we’ll make a new `path` string using the new p1 and p2, using **`p1.path|”,”|p2.path`**. Finally, we’ll sum the distances of the two paths **`p1.dist + p2.dist`**.

```
select p1.start, p2.finish, p1.path|”,”|p2.path, p1.dist + p2.dist from
candidates as p1, G as p2 where p1.finish = p2.start and p1.dist + p2.dist
< 300 and finish != “spaceship”
```