Hungarian Algorithm | Ride Share Allocation

Introduction

Zippy-Roo Ride Share company has five riders that have just registered for rides: Angela, Belinda, Christine, David and Erin. Wait times per vehicle (C_1, C_2 … C_5) to reach each rider are as follows:

- Angela = {6, 5, 8, 4, 8}
- Belinda = {5, 4, 3, 8, 9}
- Christine = {4, 6, 8, 7, 7}
- David = {7, 9, 6, 4, 6}
- Erin = {3, 4, 5, 6, 5}

Zippy-Roo do not have a car pool option, so only one rider can be allocated to each vehicle. Vehicles (C_1, C_2 … C_5) could be allocated immediately as riders register, however, this is unlikely to generate the optimal solution for riders or drivers. Consider for example that Angela gets Car 1 (C_1), Belinda Car 2 (C_2) … Erin Car 5 (C_5). The *total* wait time would be: 6 + 4 + 8 + 4 + 5 = 27 minutes. An immediate improvement would be to assign Car 1 to Christine and Car 3 to Angela. There are so many other ways of performing the allocation, an algorithm that automatically minimises the overall wait time would be useful.

You may have noticed that when you or a friend register for a ride, the App searches for an available driver. An algorithm is busy identifying the best solution or allocation. Zippy-Roo's goal is to minimise the total amount of wait time. The Hungarian algorithm is a relatively simple and accessible algorithm.

Hungarian Algorithm

If you have not used the Hungarian Algorithm before, scan the QR code opposite to watch a video to help:

- Understand how and what the algorithm does
- Apply the algorithm to solve an allocation problem
- How to solve it on your calculator.

You can download the algorithm from the Texas Instruments Australia website: http://education.ti.com/Australia

Navigate to the Student section, select: Student Resources > Tutorial Series > Tutorials by State & Subject, then select your course.

To see how to install the widget, scan the QR code opposite:

Step 1 – Setting up the Problem

Express the problem information in a matrix and store it as a variable.
(Example: M := …)

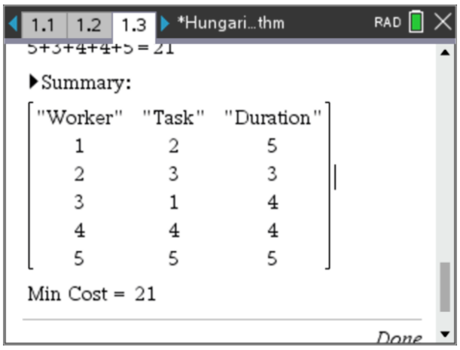
> When entering values in the matrix, use **tab** to navigate.

Step 2 – Generating a Solution

Insert the Hungarian Algorith Widget, follow the instructions and run the program.

Author: Peter Fox
Widget: LazyMath

Solution:



Step 3 – Interpreting the solution

Worker = Rider

Task = Car Allocation

Duration = Time

Therefore:

- Angela (Worker 1) is assigned to Car 2 (Task 2) which takes: 5 minutes;
- Belinda (Worker 2) is assigned Car 3 (Task 3) which takes 3 minutes;
- Christine (Worker 3) is assigned Car 1 (Task 1) which takes 4 minutes;
- David (Worker 4) is assigned Car 4 (Task 4) which takes 4 minutes;
- Erin (Worker 5) is assigned Car 5 (Task 5) which takes 5 minutes.

Overall time: 5 + 3 + 4 + 4 + 5 = 21 minutes.

Retrieval vs Encoding

In a UK study, students attended a lecture on a specific topic for which they would later be examined. The students were divided into two groups and given the same amount of time to study and prepare:

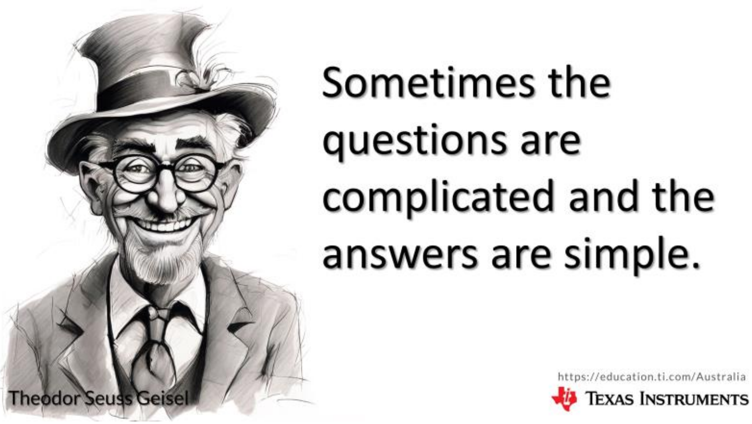
Group A: These students were provided with notes from the lecture, videos and loads of reading.

Group B: These students answered lots of questions.

Group A went into the exam quite confident they had learned and remembered everything. Group B were much less confident, however, they scored, on average, 20% points better than Group A. Why?

Group A continued to encode. Group B were retrieving. Your brain is not a hard drive, you need to do so much more than write (encode) data into your brain, you must spend significant amounts of time retrieving the information. In mathematics, that mostly equates to doing questions.

Check out our other worksheets for more study tips!



Author: Peter Fox