CS2010 PS7 - A Trip to Supermarket v3 (with R-option)

Released: Thursday, 24 October 2013, 00.00am (Happy 2nd Birthday for Jane)
Due: Saturday, 09 November 2013, 8am

Collaboration Policy. You are encouraged to work with other students or teaching staffs (inside or outside this module) on solving this problem set. However, you **must** write the Java code **by yourself**. In addition, when you write your Java code, you **must** list the names of every collaborator, that is, every other person that you talked to about the problem (even if you only discussed it briefly). This list may include certain posts from fellow students in CS2010 IVLE discussion forum. Any deviation from this policy will be considered cheating, and will be punished severely, including referral to the NUS Board of Discipline. It is not worth it to cheat just to get 15% when you will lose out in the other 85%.

R-option. This PS has R-option at the back. This additional task usually requires understanding of data structure or algorithm *beyond* CS2010. A self research on those relevant additional topics will be needed but some pointers will be given. CS2010R students *have to* attempt this R-option. CS2010 students can choose to attempt this R-option too for higher points, or simply leave it.

2011 Story. Before Jane's birth, Grace could not travel far and could not carry heavy stuffs (see PS4). After Jane's birth, Grace needs a lot of time to rest and to breastfeed Jane. Therefore, Steven has to help Grace with a certain household chore that he thought he does not need to do anymore after he got married: Grocery Shopping. (To the guys reading this, get ready :D). Note that although Jane is now 2 years old, it is still much easier to do grocery shopping *without* bringing her to the supermarket. So, Steven actually has not been able to escape from this chore.

The Actual Problem. Today, Steven has to visit a supermarket at Clementi area to buy groceries (name omitted to avoid indirect advertising). Steven has visited this place numerous times and therefore he knows the location of various N items in that supermarket. Steven has estimated the *direct* walking time from one point to every other points in that supermarket (in seconds) and store that information in a 2D table `T` of size $(N+1) \times (N+1)$. Given a list of K items to be bought today, he wants to know what is the minimum amount of time to complete the shopping duty of that day. As mentioned several time in lectures, we have to make some simplifying assumptions in order for this problem to be solvable... :

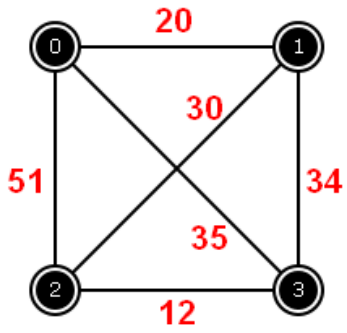
1. Steven always starts at vertex 0, the entrance (+ cashier section) of that supermarket.
2. There are $V = N + 1$ vertices due to the presence of this special vertex 0.
   The other N vertices corresponds to the N items. The vertices are numbered from `[0..N]`.
3. The direct walking time graph that is stored in a 2D table `T` is a **complete graph**.
   `T` is a symmetric square adjacency matrix with `T[i][i] = 0` $\forall$`i` $\in$ `[0..N]`.
4. Steven has to grab all K items (numbered from `[1..K]`) that he has to buy that day, or he will have hard time explaining (to Grace) why he does not buy certain items... $1 \leq K \leq N$.

5. Steven is very efficient, once he arrives at the point that stores item i, he can grab item i into his shopping bag in 0 seconds (e.g. for item 'banana', he does not have to compare the price of 'banana A' versus 'banana B' and he does not have to select which banana looks nicer, etc...). So, Steven's shopping time is only determined by the total walking time inside that supermarket to grab all the K items.

6. In this problem, Steven ends his shopping duty when he arrives at cashier (also at vertex 0) after grabbing all the required K items.

7. Steven can grab the K items in **any order**, but the total walking time (i.e. time to walk from vertex 0 → to various points in the that supermarket in order to grab all the K items → back to vertex 0) **must be minimized**. Steven can choose to bypass a certain point that is *not in his shopping list* or even *revisit* a point that contains item that he already grab (he does not need to re-grab it again) if this leads to faster overall shopping time.

See below for an example supermarket:



	0	1	2	3
0	0	20	51	35
1	20	0	30	34
2	51	30	0	12
3	35	34	12	0

2D Table T

Figure 1: A Sample Supermarket Layout, $N = 3$, $V = 3 + 1 = 4$

If today, Steven has to buy all item 1, item 2, and item 3, then one of the best possible shopping route is like this: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$ with a total walking time of: 20+30+12+35 = 97 seconds.

If tomorrow, Steven has to buy only item 1 and item 2, then one of the best possible shopping route is still: $0 \rightarrow 1 \rightarrow 2(\rightarrow 3) \rightarrow 0$ with a total walking time of: 20+30+(12+35) = 97 seconds. Notice that although Steven does not have to buy item 3, taking sub path $2 \rightarrow 3 \rightarrow 0$ (where Steven will just bypass item 3) is faster than taking sub path $2 \rightarrow 0$.

If two days later, Steven has to buy only item 2, then one of the best possible shopping route is like this: $0(\rightarrow 3) \rightarrow 2(\rightarrow 3) \rightarrow 0$ with a total walking time of: (35+12)+(12+35) = 94 seconds. Steven bypass through item 3 twice!

If three days later, Steven has to buy only item 2 and item 3, then one of the best possible shopping route is like this: $0 \rightarrow 3 \rightarrow 2(\rightarrow 3) \rightarrow 0$ with a total walking time of: 35+12+(12+35) = 94 seconds. See that Steven can revisit point 3 although he does not have to grab item 3 twice.

These four examples should be clear enough to describe this problem.

The skeleton program `Supermarket.java` is already written for you, you just need to implement one (or more) method(s)/function(s):

- `int Query()`
  You are given a 2D matrix `T` of size $(N + 1) \times (N + 1)$ and an array `shoppingList` of size K. Query these two data structures and answer the query as defined above.

- If needed, you can write additional helper methods/functions to simplify your code.

Subtask 1/Problem A in Mooshak (20 points). The supermarket is a very small convenience store and everything there have to be grabbed/bought. $1 \leq K = N \leq 9$.

Subtask 2/Problem B in Mooshak (Additional 20 points). The supermarket is still a very small convenience store but slightly bigger than in Subtask 1 and everything there have to be grabbed/bought. $1 \leq K = N \leq 15$.

Subtask 3/Problem C in Mooshak (Additional 20 points). The supermarket has the same size as in Subtask 2, but only a subset of K out of N items have to be grabbed/bought. $1 \leq K \leq N \leq 15$.

Subtask 4/Problem D in Mooshak (Additional 20 points). The supermarket is a minimart that sells $\approx$ *hundreds* of grocery items. However, Grace only asks Steven to buy some items. $1 \leq N \leq 200$, $1 \leq K \leq 15$, $K \leq N$.

Subtask 5/Problem E in Mooshak (Additional 20 points). The supermarket is a large supermarket that sells $\approx$ *thousands* of grocery items. However, Grace only asks Steven to buy some items. $1 \leq N \leq 1000$, $1 \leq K \leq 15$, $K \leq N$.

Subtask 6/Problem F in Mooshak (Additional 10 points)/R-option. The supermarket is a large supermarket that sells $\approx$ *thousands* of grocery items. However, Grace only asks Steven to buy some items. $1 \leq N \leq 1000$, $1 \leq K \leq 19$, $K \leq N$ (Notice the new range of K).

WARNING: This Subtask 6 requires something that I do not teach in CS2010 and can be very frustrating (lots of TLE) if you do not use 'that technique(s)'. CS2010 students, please attempt this with caution. CS2010R students, you need to figure out what is/are the required technique(s) to solve this problem. To simplify the grading process because we are nearing the end of the semester, we will simply give 0 point (no partial points) for any non accepted solution to problem F in Mooshak.

Note: The official test data has been uploaded to Mooshak online judge, but it is hidden from your view. The time limit setting in Mooshak online judge for Subtask 1,2,3,4,5,6 are 1,1,1,1,3,7 seconds, respectively. You are encouraged to generate and post additional test data in IVLE discussion forum. Please use `SupermarketVerifier.java` to verify whether your custom-made test data conform with the required specifications. This program will test if:

1. The shopping items are within [1..N].
2. The given matrix contains non-negative integers, symmetric, and all entries in the main diagonal of the matrix are zeroes.

Figure 2: Jane's 1st Birthday, 24 October 2012