CS2010 PS7 - A Trip to Supermarket

Released: Thursday, 20 October 2011
Due: Saturday, 29 October 2011, 8am

Collaboration Policy. You are encouraged to work with other students on solving this problem set. However, you **must** write up your solution **by yourself**. In addition, when you write up your solution, 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). 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%.

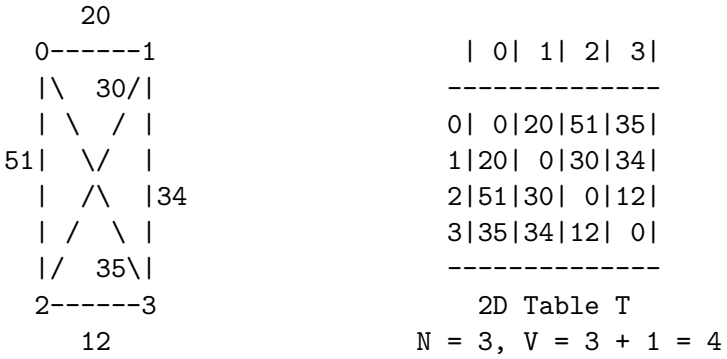
The Story. Technically, Jane should have already been born by the time this PS is due (29 October 2011) and Steven will take lots of holiday leave during the days when he is not teaching. Since Grace needs a lot of time to rest and to breastfeed Jane, Steven needs to help Grace with various household chores that he has not done that often since he got married. One of the task that he has to do is: Grocery Shopping.

This time, Steven has to visit a supermarket at Clementi area (name omitted to avoid indirect advertising). Steven has visited this place several times and therefore he knows the location of various N items in that supermarket. Steven has even 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.

The Actual Problem. 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, 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).
7. Steven can grab the K items in **any order**, but the total walking time (i.e. time to walk from vertex 0 $\rightarrow$ to various points in the that supermarket in order to grab all the K items $\rightarrow$ 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:



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 $2 \rightarrow 3 \rightarrow 0$ that 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.

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.

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. The entries in `shoppingList` are sorted in ascending order.
- If needed, you can write additional helper methods/functions to simplify your code.

Subtask 1 (50 points). The supermarket is a small supermarket and everything there have to be bought. $1 \leq K = N \leq 9$.

Subtask 2 (25 points). The supermarket is slightly bigger than in Subtask 1 and everything there have to be bought. $1 \leq K = N \leq 15$.

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

Subtask Bonus 1 (15 points). Typical supermarket size that sells $\approx$ *hundreds* of grocery items. However, Grace only asks Steven to buy some items. $1 \leq N \leq 199$, $1 \leq K \leq 15$, $K \leq N$.

Subtask Bonus 2 (5 points). Typical supermarket size that sells $\approx$ *thousands* of grocery items. However, Grace only asks Steven to buy some items. $1 \leq N \leq 999$, $1 \leq K \leq 15$, $K \leq N$.

Note: The test data to reach 50 points: `Subtask1.txt` and `Subtask1-ans.txt` are given to you. You are allowed to check your program's output with your friend's.