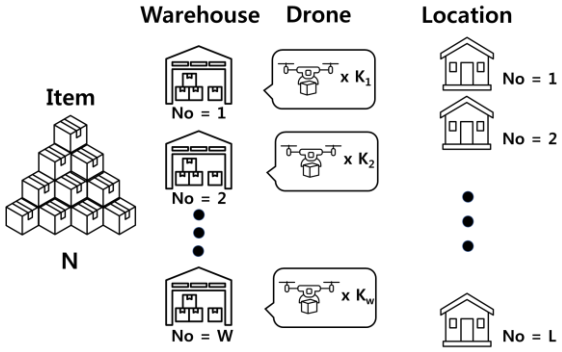
Algorithms Homework Assignment #6

"Drone Package Delivery" (3+1 points)

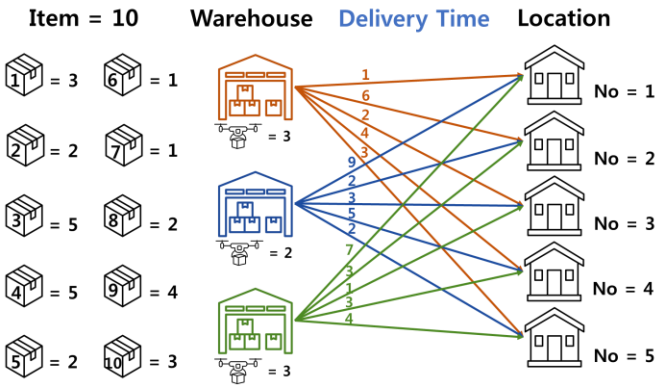
Due date: Monday, June 11th 2018, 9AM.
- Submit a report and source code on the server.

In this homework assignment, you'll design and implement an algorithm for "package delivery". Specifically, your algorithm will compute a 'warehouse assignment' for package delivery such that you can minimize the finishing time of all delivery. This is an open-ended problem: I do not have an optimal solution to this, and whoever designs the best algorithm (whoever generates the best assignment with best delivery time) will get the best score.



Let's first define and declare a few things in this problem;

- There are W = 3 warehouses, W_1, W_2, W_3.
  - We will simply write {W} = { 1, 2, 3 } for simplicity.
- Each warehouse has K_W drones, K_1, K_2, K_3.
  - These drones (at each warehouse) will be used for package delivery from that warehouse.
  - Drones do not go to other warehouses. They only deliver to 'locations'.
- There are L = 5 locations, L_1, L_2, L_3, L_4, L_5.
  - These are potential locations for package destination. You need to deliver to one of these locations.
  - We will simply write {L} = { 1, 2, 3, 4, 5 } for simplicity.
- You will be given a table for the time it takes to deliver a package from a warehouse W_i to a location L_j.
  - That is, you will be given a W * L = 3 * 5 = 15 values for the delivery time in a matrix form;
  - T[i][j] = time to deliver a package from a warehouse W_i to a location L_j.
  - T[i][j] is a round-trip-time. (this is shown as 'delivery time' in below figure)
- You will be given N items to deliver, {$item_1$, $item_2$, $item_3$, … $item_i$, ... $item_N$}.
  - For simplicity, we will write {ITEM} = { 1, 2, 3, … k, … N }.
  - Each item has a specified destination D[k] for each item k.
    - One of the location L_j is the destination of item k.
    - D[k] = L_j for all k in N, where j is given as input. (shown on the left side of below figure)



You GOAL is the minimize the time $T_{min\text{-}total}$ required to finish all package delivery.

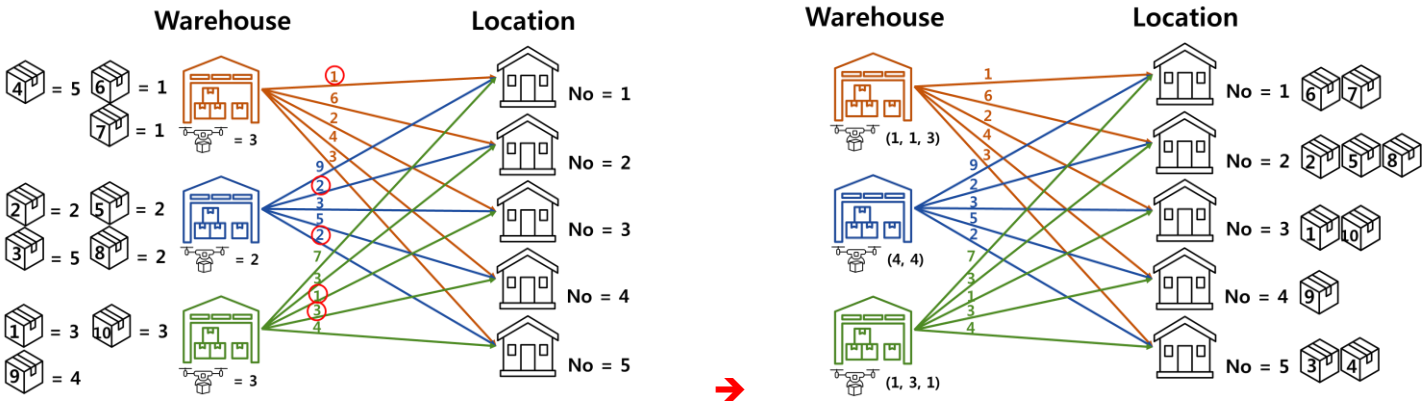
- For each item i, you need to decide which warehouse W_j it should be delivered from.
  - Let the warehouse assignment for item i, A[i] = W_j.
  - Then you need to decide W_j. W_j is the warehouse from which item i will be delivered.
- Once you have decided on all the assignments, calculate the $T_{min\text{-}total}$.
  - Note that the time $T_{min\text{-}total}$ required to finish all package delivery is the time that the delivery of last package finishes.

For example, you might assign warehouses to items like left figure below;

➔ then, item will be delivery like right before below, where

- warehouse 1 finishes delivery of 3 packages using 3 drones in t = 3
- warehouse 2 finishes delivery of 4 packages using 2 drones in t = 4
- warehouse 3 finishes delivery of 3 packages using 3 drones in t = 3

➔ thus, results in final delivery time $T_{min\text{-}total}$ = 4.



There are many numbers, so please be careful not to confuse

1) 'item number' ($item_k$),
2) 'destination location number of each item' (D[k] = L_j),
3) 'warehouse number' (W_i = i), 'number of warehouses' (W),
4) 'number of drones at each warehouse' (K_i),
5) 'delivery time (distance) between each warehouse and location' (T[i][j]),
6) 'location number' (L_j), 'number of locations' (L),
7) 'starting and ending time of package delivery' (whatever you calculate), and
8) 'final finishing time of all delivery' ($T_{min\text{-}total}$).

Important constraint:

- No more that K_i items can be in delivery simultaneously from a warehouse W_i.
  - For example, if a warehouse has 3 drones, 3 drones can fly out for delivery of 3 packages, but 4^{th} package must wait until one of the drones come back to the warehouse.
- Also note that the delivery time T[i][j] may be different for each location L_j. Thus, the return time of the drone can be different depending on which location it is delivering to.
  - Recall time T[i][j] is the round-trip delivery time from warehouse i to location j.

Simplifying Assumptions:

- Assume that each warehouse has enough items to satisfy all delivery requests.
- Assume that a drone has infinite battery so that it does not need to recharge.
  - A drone can deliver another package as soon as it finishes delivery of previous package.
- Assume that a drone will not complain or break or strike even if you give too much work to do.
- Assume that input file is correct (there is no error in the input file)

Your task is to

1. Design and implement an algorithm to compute a 'warehouse assignment' that meets all the requirements.
   - ✓ That is, for each and every item, determine which warehouse it should be delivered from.
2. Print out on the screen the warehouse assignment, as well as the final finishing time.
3. Write a short report on how you approached the problem and what your idea is on solving the problem.

Input file format will be as follows;

- It has W and L in the first two lines, followed by
- [W lines of] warehouse number, number of drones, and a row of distance matrix T[i][j] in each line
- N, the number of items, and
- [N lines of] item number, and the destination location number in each line
- Below is an example of an input file (, not including the red comment part);

```
$ cat testinput/input-3-5-10-1.txt
3                       // W, number of warehouse
5                       // L, number of locations
1 2 1 6 1 6 4           // W1, K1, T[1][j] for j=1,…,5
2 1 6 2 3 6 9           // W2, K2, T[2][j] for j=1,…,5
3 1 7 3 1 5 1           // W2, K3, T[3][j] for j=1,…,5
10                      // N, number of items
1 3                     // item1, D[1] = L3
2 2                     // item2, D[2] = L2
3 5                     // item3, D[3] = L5
4 5                     // …
5 2
6 1
7 1
8 2                     // …
9 4                     // item9, D[9] = L4
10 3                    // item10, D[10] = L3
$
```

Screen output format **MUST** be as follows; (it must be exact, otherwise it cannot be graded)

- N, the number of items, and
- [N lines of] item number, and warehouse number of your choice
- Total delivery time = <T>
- Below is an example of a screen output (, not including the red // comment part);

```
$ ./package_delivery input-3-5-10-1.txt
10                      // N
1 1                     // item1, from W1, to L3, finish at t=1
2 2                     // item2, from W2, to L2, finish at t=2
3 3                     // item3, from W3, to L5, finish at t=1
4 3                     // item4, from W3, to L5, finish at t=2
5 2                     // item5, from W2, to L2, finish at t=4
6 1                     // item6, from W1, to L1, finish at t=1
7 1                     // item7, from W1, to L1, finish at t=2
8 3                     // item8, from W3, to L2, finish at t=5
9 1                     // item9, from W1, to L4, finish at t=7
10 1                    // item10, from W1, to L3, finish at t=3
Total delivery time = 7
$
```

Your total delivery time must be VALID and POSSIBLE with your warehouse assignment!
The ordering of delivery may be different, however. For example, you may deliver item 10 before item 9.

You must implement using **C**, and you must implement a program "package_delivery.c". The file names must be exactly as given, otherwise it won't be graded and you will get zero points.

Compile your programming using gcc on linux

- Ex> `gcc -Wall -o package_delivery package_delivery.c`

Your program must take two command line arguments: <input filename> and <N>

- Ex> `./package_delivery input-3-5-10-1.txt`

You program must work with other input files as well, not just '`input-3-5-10-1.txt`'. We will test and grade your program with other input files with different number of warehouses / locations / items, and different destinations for each item.

Report Document

- You need to write a short report document for this homework assignment. The report should NOT include your source code. The report should include your idea and approach on tackling this problem, and your algorithm in your own words. It may include a pseudo code. Better would be to include a figure or a diagram that explains your algorithm.
- You can write either in Korean or English (sorry, no French nor Spanish nor Chinese).
- No limit on the length, but not too long please.
- Any other information needed to compile and run your program.
- Please use Microsoft Word (.docx) for the report.

What and how to submit

- You must submit softcopy of all the files, including the report file and the source code;
- Here is the instruction on how to submit the softcopy files :
  ① Login to your server account at nsl2.cau.ac.kr.
  ② In your home directory, create a directory "submit_alg/submit_<student ID>_hw6" (ex> "/student/20149999/submit_alg/submit_20149999_hw6")
  ③ Put your source files in that directory, together with a Makefile that can compile your code by simply typing 'make'. Also, put your report (in MS .docx file) in the same directory.
- Your submission will not be graded if you do not follow the instructions exactly.

Other requirements:

- Your program must run on our class Linux server at nsl2.cau.ac.kr.
- Your code must include your name and student ID at the beginning of the code as a comment.
- Your code should be easily readable and include sufficient comments for easy understanding.
- Your code must be properly indented. Bad indentation/formatting will result in score deduction.
- Your code should not include any Korean characters.
- Your program should work regardless of whether the input file format is Windows/Linux/MacOS based.
  ✓ That is, input file may have Windows or Linux or MacOS line ending format (\n, \r\n, \r).

Grading criteria:

- You get 3 points
  ✓ if your program works as required, AND you meet all of above requirements, AND
  ✓ if your report is well written.
- Otherwise, partial deduction may apply.
- You may get up to extra 1 point
  ✓ if your program generates a valid & fast package-to-warehouse assignment that is better than others.
  ✓ Better delivery time will get better score.
- No delayed submissions are accepted.
- Copying other student's work will result in negative points.
- Code that does not compile or code that does not run will result in negative points.