300103
Data Structures and Algorithms
Assignment 1
Submit Deadline: 5:00pm 15th September 2011

1. Problem: Train Reorganising

A freight train from Melbourne is approaching Sydney, carrying n cars of cargos. The cargos are to be delivered to n different cities in the metropolitan area of Sydney - one car for one city. To avoid delays of stoping at each city station, the train is to dismiss a car once the car arrives its destination station without unloading the car, providing the car was at the end of the train. Since all these cars of the train were collected on the way from Melbourne to Sydney, the order of these cars does not match the order of their designations. The train have to be reorganised in the Railway Switching Junction outside Sydney to reorder its cars to match the order of their destinations (See Figure 1). The aim of this assignment is to design and implement an algorithm to simulate the procedure of car re-ordering.

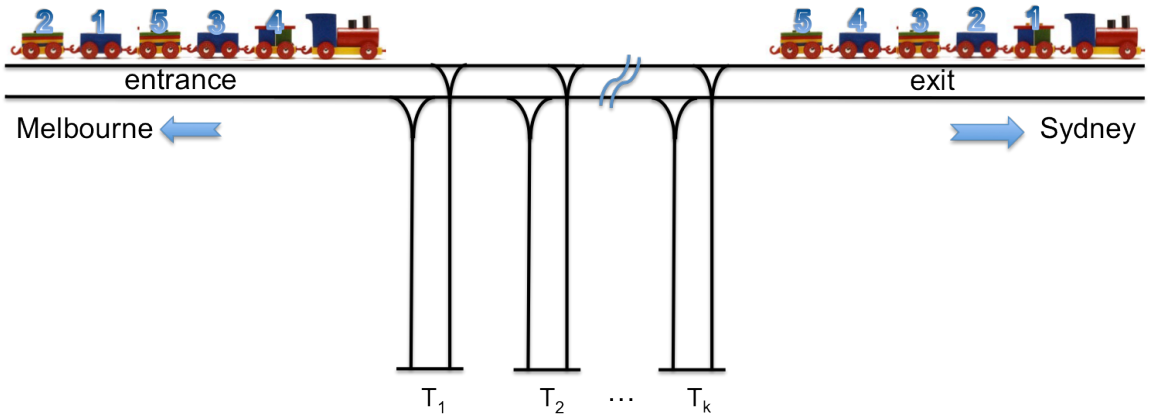

Figure 1: Sydney Railway Switching Junction

The above figure shows the diagram of the railway switching junction. The junction has a single entrance and a single exit on the main railway from Melbourne to Sydney. There are k transit rails, denoted by T_1, T_2, …, and T_k, linked to the main railway for a train to transfer cars. The train can drive in or out each transit rail either forward or backward. The train can be disconnected at any point between two cars. A car can be parked in a transit rail but cannot be left on the main track. A car cannot move without connecting to the engine (only one engine). We assume that the transit rails and their distance in between are long enough for the train to drive in and out in full length.

Assume that the destination stations in Sydney are labelled from 1 to n, arriving Station n first and ending at Station 1. Each car of the train is marked in accord with its destination. When the train arrives, the order of the cars from the engine to the last car is $c_1,c_2,\ldots,c_n$, which can be any permutation of the numbers 1,2,…,n. As an example shown in Figure 1, when the train arrives the junction, the order of the cars is 4,3,5,1,2. The cars need to be re-ordered into 1,2,3,4,5 to be delivered at their destination stations.

2. Program tasks and specification

Design and implement an algorithm to simulate car re-organizing of the train at the railway switching junction. You can only use stacks as the data structure to represent the train and the cars in each transit rail.

Task 1: For any given input of car order is $c_1,c_2,\ldots,c_n$ and a number k of transit rails ($k\geq 2$), design and implement an algorithm to simulate the car ordering procedure by using stack operations only so that the output of the car order is 1,2,…,n.

Task 2: Analyse efficiency of your algorithm using Big-O notation by counting the number of stack operations used in your algorithm (worst case analysis).

Task 3: Assume that k=n/2+1. Improve your algorithm so that its complexity is in O(n).

Task 4: Discuss the efficiency of your algorithm in relation to k if k can be any number such that $k\geq 2$.

You do not have to write your own stack ADT. You are allowed to use STL stack or the stack ADT provided in the lecture.

3. Marking criteria

Grade	Requirement
Pass	Complete task 1. You can fix n to be 5 and k to be 2. Text message to display cars' transformation.
Credit	Complete tasks 1 and 2 for any n and k. Visualize car's transformation using text-based output. A brief description of algorithm analysis (within one A4 page).
Distinction	Complete tasks 1-3. Visualize car's transformation using text-based output. A brief description of your algorithm and an analysis of its complexity (within two A4 pages).
High Distinction	Complete tasks 1-4. Visualize car's transformation using text-based output. Detailed analysis of algorithm efficiency (within four A4 pages).

No matter which level of program you have implemented, your program should be executable. No incomplete program is acceptable.

4. Deliverables

4.1 Source code

You are only allowed to use C++ to code your solution. You can use any C++ compiler or IDE to demonstrate your program provided it is available during your demonstration. You are allowed to demonstrate your program on your laptop. **All comments must be deleted from your source code when you demonstrate.** The code should be purely written by you. No part of the code can be written by any other persons or copied from any other source except for the Stack ADT.

4.2 Declaration

All students are required to submit a document contain the following

DECLARATION

I hereby certify that no part of this assignment has been copied from any other student's work or from any other source. No part of the code has been written/produced for me by another person or copied from any other source.

I hold a copy of this assignment that I can produce if the original is lost or damaged.

4.3 Algorithm description and efficiency analysis

In addition to the declaration, the students who seek for Credit or higher are required to submit a document for algorithm analysis. The document should be formatted in Word and submitted to vUWS in Word or PDF.

5. Submission

Both the documentation and source code should be submitted via vUWS before the deadline for documentation purpose. Your programs (.h, .cpp) can be put in separate files (executable file is not required). All these files should be zipped into one file **with your student id as the zipped file name**. Submission that does not follow the format will not be accepted.

Email submission is not acceptable (strict rule).

6. Demonstration

You are required to demonstrate your program during **your scheduled** practical session between 19-23 Sep 2011. Your tutor will check your code and your understanding of the code. Contact your tutor for booking your demonstration time. **You will receive no**

marks if you fail the demonstration, especially if you fail the demo booking or miss the demo time. Note that it is students' responsibility to get the appropriate compilers or IDEs to run their programs. You are allowed to run your program from your laptop at your booked demo time. **The feedback to your work will be delivered orally during the demonstration.** No further feedbacks or comments are given afterward. You are highly recommended to hand in a hard copy of your documentation to your tutor during the demonstration so that your tutor could have a better understanding of your algorithm. In any case you must submit a softcopy of your document to vUWS by the deadline.

The demonstration program should be the same as the one you submit except that all the comments should be taken off during the demonstration. If you fail this assignment at your first demonstration, you are allowed to improve your work within one week (**maximal grade is pass in this case**).

Do not send your work to the unit coordinator. Your tutor is responsible to mark your work.