Assignment 6

Due date: 10:00am, Wednesday, the 26th of October 2016.
This assignment is worth of 6% of the total course mark.

General Instructions

You have to do it **individually**.

Paper submissions have to include coversheet including names, student ids, and your tutorial groups such that submissions can get marked.

For Programming submissions upload your files in websubmission **by November 2, 10:00am**. **No late submissions will be accepted.**
Names and IDs of group members must be specified in a comment in the beginning of all files that you submit.

Exercise 1 *(60 points) Programming Exercises*

For all of the following exercises, write a program, test and and submit the file in the web submission system. Your program should have a main function and inputs such as the path of the graph file are passed as arguments for the main function.

1. Network bottlenecks are places (edges) in the telecommunication networks which are slow due to any technical issues. They slow down the network speed of the users who are using that network. Implement an algorithm (in a file called BottleNeck.java) that receives a bidirectional graph $G = (V, E)$ as an adjacency matrix, where $E_{i,j} = E_{j,i}$ and $E_{i,j} \in \mathtt{N}^+$ denotes an existing edge between i and j with the cost $E_{i,j}$ iff $E_{i,j} > 0$. However, given the graph G, the edge with the maximum cost may not be a bottleneck edge if a shorter path between its node could be found. Use Minimum Spanning Trees in your program to find out the edges which are bottlenecks of the network. As the output, your program should print the indexes of the two nodes for each bottleneck edge. *[30 marks]*

2. Design and implement a function (in a file WordSearchPuzzle.java) that takes in two input. The first input is the path of the word puzzle file and the second input is the path of word list. A word search puzzle consists of an $n \times n$ grid of characters and a "word list" containing m words each length l. Each of the m words in the word list is hidden in the grid, either horizontally or vertically (not diagonally in this problem). Note that the words may be hidden in reverse, as seen in the case of SORT and DICT in the example below. Using the hash table concept, write a program that finds all the words in the word bank in the puzzle (i.e., prints the

coordinates of the first letter of each word, based on the order of the words given). In the output, your program should print three details. First, the number of all collisions in the hash table. Secondly, it should print out in each line the original word, and the indexes of the first letter (row, col) separated by comma. To be efficient, you may assume that $l \leq 5$. *[30 marks]*

Table 1: Word Search Example ($n = 8$, $m = 6$, $l = 4$

H	T	A	Q	**A**	C	O	**T**
A	R	D	L	**L**	W	I	**C**
S	**M**	I	L	**G**	R	S	**I**
H	**A**	B	O	**O**	A	H	**D**
X	**T**	T	J	I	E	H	D
N	**H**	B	M	**H**	**E**	**A**	**P**
U	X	T	O	O	C	D	W
F	Q	**T**	**R**	**O**	**S**	C	K

Word bank: ALGO, DICT, HASH, HEAP, MATH, SORT

Paper Based Questions

Exercise 2 *Hashing (10 points)*

You are given a hash table with n keys and m slots, with the simple uniform hashing assumption (each key is equally likely to be hashed into each slot). Collisions are resolved by chaining.

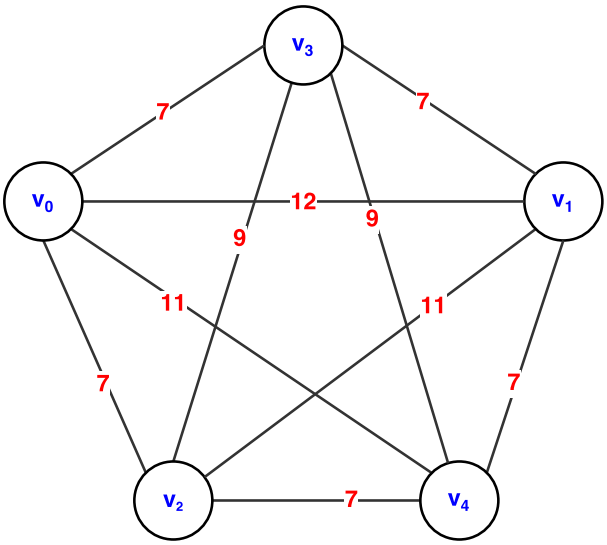
1. What is the probability that the first slot ends up empty?
2. What is the expected number of slots that end up not being empty?

Exercise 3 *Hashing Application (15 points)*

Let S and T be two sets of numbers, represented as unordered linked lists of distinct numbers. All you have are pointers to the heads of the lists, but you do not know the list lengths. Describe an $O(min|S|, |T|)$-expected-time algorithm to determine whether $S = T$. Discuss the correctness of your algorithm's run-time.

Exercise 4 *Kruskal's Algorithm (15 points)*

Compute a minimum spanning for the following graph using Kruskals algorithm. Show the status of your partial minimum spanning tree after each edge insertion and indicate for each edge whether it is included in the minimum spanning tree.



Instructions for programming code

First, type the following command, all on one line (replacing `aXXXXXXX` with your username):

```
svn mkdir --parents -m "ADSA"
https://version-control.adelaide.edu.au/svn/aXXXXXXX/2016/s2/adsa/assignment-06
```

Then, check out this directory and add your files:

```
svn co https://version-control.adelaide.edu.au/svn/aXXXXXXX/2016/s2/adsa/assignment-06
cd assignment6
svn add File1.java
add File2.java ...
svn commit -m "assignment6 solution"
```

Next, go to the web submission system at:
https://cs.adelaide.edu.au/services/websubmission/
Navigate to 2016, Semester 2, Adelaide, Algorithm and Data Structure Analysis, then Assignment 6. Click Make A New Submission For This Assignment and indicate that you agree to the declaration. The script will then check whether your code compiles. You can make as many resubmissions as you like. If your final solution does not compile you would not get any marks for this solution.

End of Questions