

COMP3600/COMP6466 in 2009 – Assignment Two

(Due: 12pm Monday, October 26 **Late Penalty:** 25% per day)

Question 1.(40/50 for COMP3600 and 40/60 for COMP6466)

Given two integers n and M with $n < M$, generate an undirected weighted random graph $G = (V, E, w)$ with $|V| = n$ and $|E| = m$, assume that M is the upper bound of m , i.e., $m \leq M$, using the following pseudo-code procedure **Generate_graph** to generate a *connected, edge-weighted* graph, where *init_seed* is the initial seed of the random function, w is the edge-weight function whose value is an integer between 1 and 200.

```
Generate_graph( $G, V, E, n, M$ )
1.    $V \leftarrow \{1, 2, \dots, n\}$ ;
2.    $init\_seed \leftarrow 2009$ ; /* the initial random seed */
3.    $number\_try \leftarrow 0$ ; /* the number of tries to find a connected graph */
4.    $srand(init\_seed)$ ; /* initial random seed */
5.   while  $G$  is not connected do
6.        $E \leftarrow \emptyset$ ;
7.        $m \leftarrow 0$ ; /* the number of the edges in the graph */
8.        $count \leftarrow 0$ ; /* the number of potential edges generated so far */
9.       repeat
10.          choose two vertices randomly, e.g.,  $u$  and  $v$ ,
11.               $u \leftarrow (rand() \bmod n) + 1$ ;
12.               $v \leftarrow (rand() \bmod n) + 1$ ;
13.           $count \leftarrow count + 1$ ;
14.          if ( $u \neq v$ ) and (no edge between  $u$  and  $v$  exists) then
15.               $E \leftarrow E \cup \{(u, v)\}$ ;
16.              /*add an edge between  $u$  and  $v$  */
17.               $w(u, v) \leftarrow (rand() \bmod 200) + 1$ ;
18.              /* generate a random integer between 1 and 200 for edge  $(u, v)$  */
19.               $m \leftarrow m + 1$ ; /* the number of edges in  $G$  is increased by 1 */
18.          endif;
19.       until ( $count = M$ );
20.        $number\_try \leftarrow number\_try + 1$ ;
21.   endwhile.
```

If $count = M$ but the resulting graph G is still disconnected, then the random value in the last round will be used as the *new random seed* to run the above procedure again until the resulting graph is connected **AND** $count = M$.

We assume that a *sparse graph* is represented by **adjacency lists**, while a *dense graph* is represented by **an adjacency matrix**. The tasks are as follows.

(a) Generate a connected, weighted graph of $n = 10$ nodes and m edges that is upper bounded by $M = \lceil 1.35 * n \log_2 n \rceil$, using algorithm **Generate_graph**. List each edge and its associated weight of the generated graph G in your report: The output format is a list of $(u, v, w(u, v))$, where nodes u and v are the two endpoints of edge (u, v) and $w(u, v)$ is the weight of the edge. **Don't print any other size graph** (6/50 for COMP3600/and 6/60 for COMP6466).

(b) Implement Kruskal's algorithm for minimum spanning trees in $G(V, E, w)$. Apply this algorithm to the generated graph G in (a), where $n = 10$ and $M = \lceil 1.35 * n \log_2 n \rceil$, by finding an MST and listing the tree edges and their weights in the order they generate (7/50 for COMP3600 and 7/60 for COMP6466).

(c) Implement Prim's algorithm for minimum spanning trees in $G(V, E, w)$. Apply this algorithm to the generated graph G in (a), where $n = 10$ and $M = \lceil 1.35 * n \log_2 n \rceil$, by finding an MST and listing the tree edges and their weights in the order they generate (7/50 for COMP3600 and 7/60 for COMP6466).

(d) When $n_1 = 15$, $n_2 = 150$, $M_1 = \lceil 1.5 * n \log_2 n \rceil$ and $M_2 = \lceil 0.4 * n^2 \rceil$, consider each of the **four** different combinations of $\{n_i, M_j\}$, generate a graph $G_{i,j}$ with n_i nodes and the number of edges upper bounded by M_j , using algorithm **Generate_graph**, $1 \leq i, j \leq 2$.

Apply the above two MST algorithms on $G_{i,j}$, list the real running time (in milliseconds) of each algorithm for $G_{i,j}$, where *the timing of an algorithm* starts from its running until its finish (you must use the dept. machine **partch**, otherwise, the random value generated by your own laptop or desktops may be different). Through examining the running time and the problem size for the above two algorithms, which conclusions can you get (which algorithm outperforms the other after which conditions)? Note that the time spent for the construction of the graph itself should **not** be taken into account. **Do not print** the resulting MSTs.

(Hint: for each graph, you may run each algorithm 10 times and use the mean of the running times as the estimated running time for the graph) (20/50 for COMP3600 and 20/60 for COMP6466)

Question 2.(10/50 for COMP3600 and 10/60 for COMP6466)

A d -dimensional box with dimensions (x_1, x_2, \dots, x_d) nests within another box with dimensions (y_1, y_2, \dots, y_d) if there exists a permutation π on $\{1, 2, \dots, d\}$ such that $x_{\pi(1)} < y_1, x_{\pi(2)} < y_2, \dots, x_{\pi(d)} < y_d$.

- Show that the nesting relation is transitive (2/50 for COMP3600 and 2/60 for COMP6466).
- Describe an efficient method to determine whether or not one d -dimensional box nests inside another (2/50 for COMP3600 and 2/60 for COMP6466).
- Suppose that you are given a set of n d -dimensional boxes $\{B_1, B_2, \dots, B_n\}$. Describe an efficient algorithm to determine the longest sequence $\langle B_{i_1}, B_{i_2}, \dots, B_{i_k} \rangle$ of boxes such that B_{i_j} nests within $B_{i_{j+1}}$ for $j = 1, 2, \dots, k - 1$. Express the running time of your algorithm in terms of n and d (6/50 for COMP3600 and 6/60 for COMP6466).

Question 3.(10/60 for COMP6466 only)

Given a directed graph $G(V, E, c, d)$ and an integer $L > 0$, associated with each edge $e = \langle u, v \rangle$ in E , there is a cost $c(u, v) > 0$ and an integral delay $d(u, v) > 0$, the problem is to find a shortest path from a source $s \in V$ to a destination $t \in V$ such that the sum of the costs of the edges in the path is minimized. Meanwhile, the sum of the delays on the edges of the path is no more than L .

Devise an efficient algorithm for this problem and analyze the running time of your algorithm.

What to submit.

The program that you submitted should run if typing the following command

```
mst n
```

It will print out the following four cases.

- $n_1 = n$, $M_1 = \lceil 1.5 * n \log_2 n \rceil$, and running time of Kruskal's algorithm
- $n_2 = n$, $M_2 = \lceil 0.4 * n^2 \rceil$, and running time of Kruskal's algorithm
- $n_3 = n$, $M_3 = \lceil 1.5 * n \log_2 n \rceil$, and running time of Prim's algorithm
- $n_4 = n$, $M_4 = \lceil 0.4 * n^2 \rceil$, and running time of Prim's algorithm

You **need** submit a **hard copy** and an **electronic copy** of the following files and/or documents, and use the **exact names** in the following list.

- `makefile`
- `mst.c`
- `a2_report.pdf`

where the file `makefile` is compulsory. The shell command `make mst` must compile and link to the program `mst`. Notice that all file names must be exactly as the same as given in the above. If you have multiple files, you may put them as subroutines (functions or procedures) and packed them into a single file.

The submission command is: `submit comp3600 ass2 file_names`