Reading from a File
Creating an Adjacency List from an Edge List

The goal of this assignment is to write the **R** function `edge2adj.r` that constructs an adjacency list for a graph given the list of edges in the graph.

Input: *myfilename*, a text string that contains the name of the file containing the list of edges of a graph.

Output: *adjlist*, an **R** list of all the edges of a graph, in which the i^{th} component of the list contains a vector of all the vertices adjacent to edge i.

For example, here is a graph:

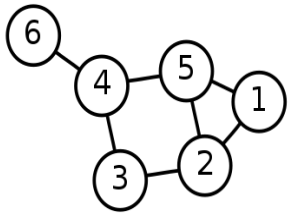

Figure 1: Graph with Six Vertices

The edge list associated with this graph is shown in Figure 2.

1	2
1	5
2	3
2	5
3	4
4	5
4	6

Figure 2: Edge List Associated with Figure 1

We want to turn this edge list into the associated adjacency list shown in Figure 3.

The first component contains the numbers 2 and 5, because Vertex 1 is adjacent to Vertices 2 and 5. The second component contains the numbers 1, 3, and 5 because Vertex 2 is adjacent to Vertices 1, 3, and 5. Etc.

There are several new **R** commands that you will need for this assignment.

1. **read.table**. Table-format files are best thought of as plain-text files with three key features that fully define how **R** should read the data.

1	2	5	
2	1	3	5
3	2	4	
4	3	5	6
5	1	2	4
6	4		

Figure 3: Adjacency List Associated with Figure 1

(a) **Header**. If a *header* is present, it's always the fist line of the file. This optional feature is used to provide names for each column of data. When importing a file into **R**, you need to tell the software whether a header is present so that it knows whether to treat the first line as variable names or, alternatively, observed data values.

(b) **Delimiter**. The *delimiter* is the character that is used to separate the entries in each line. The delimiter cannot be used for anything else in the file. This tells **R** when a specific entry begins and ends.

(c) **Missing Values**. This is another unique character string used exclusively to denote a missing value. When reading the file, **R** will turn these entries into the form it recognizes: `NA`.

Here's the plain-text file `MyFirstDataFile.txt`.

```
name age.years gender entertaining age.months
Peter * M High 504
Lois 40 F * 480
Meg 17 F Low 204
Chris 14 M Med 168
Steve 1 M High *
Brian * M Med *
```

Using the **read.table** command, **R** will now read this in as a matrix.

```
> mydatafile <- read.table(file = "MyFirstDataFile.txt", header = TRUE,
                           sep = ' ', na.strings = "*",
                           stringsAsFactors = FALSE)
> mydatafile
   name age.years gender entertaining age.months
1 Peter        NA      M         High        504
2  Lois        40      F           NA        480
3   Meg        17      F          Low        204
4 Chris        14      M          Med        168
5 Steve         1      M         High         NA
6 Brian        NA      M          Med         NA
> mydatafile[1,2]
[1] NA
> mydatafile[2,3]
[1] "F"
> mydatafile[1,4]
[1] "High"
> mydatafile[2,2]
[1] 40
> mydatafile[2,2] + 3
[1] 43
```

Note that text strings are read as text strings and that numerical values are read as numbers here.

For this assignment, there will be no header, the delimiter will be a space, and `na.strings` and `stringsAsFactors` do not have to be in the command.

2. **nrow**. We will need to know how many rows are in the matrix `E` that contains the list of edges. Use the **R** command `nrow`. Here is an example using the matrix `mydatafile`.

```
> nrow(mydatafile)
[1] 6
```

3. **max**. We will need to know how many vertices are in the graph that is represented by the list of edges. Use the **R** command `max`. Here is an example using the set $\{1, 5, 3, 7, 10, 9, 4, 6\}$.

```
> max(c(1,5,3,7,10,9,4,6))
[1] 10
```

4. **list**. To build an adjacency list, we first must initialize a list with empty components. Here's the way to initialize a list of ten components, each of which contains the empty set in **R**.

```
> Adj.List <- list(NULL)
> Adj.List
[[1]]
NULL

> Adj.List[[11]] <- 0
> Adj.List
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

[[5]]
NULL

[[6]]
NULL

[[7]]
NULL

[[8]]
NULL

[[9]]
NULL

[[10]]
NULL
```

```
[[11]]
[1] 0

> Adj.List[11] <- NULL
> Adj.List
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

[[5]]
NULL

[[6]]
NULL

[[7]]
NULL

[[8]]
NULL

[[9]]
NULL

[[10]]
NULL
```

Here are several examples.

```
> edge2adj("Edgelist01.txt")
[[1]]
[1] 2 3 4

[[2]]
[1] 1 3 7

[[3]]
[1] 1 2 4 6

[[4]]
[1] 1 3 5

[[5]]
[1] 4 6 9

[[6]]
[1] 3 5 7 8 9

[[7]]
[1] 2 6 8

[[8]]
[1]  6  7  9 10

[[9]]
[1]  5  6  8 10

[[10]]
[1] 8 9

> edge2adj("Edgelist02.txt")
[[1]]
[1] 2 7 9

[[2]]
[1]  1  5  7 10
```

```
[[3]]
[1] 5 7

[[4]]
[1] 5 6 9

[[5]]
[1]  2  3  4  6 10

[[6]]
[1] 4 5

[[7]]
[1] 1 2 3

[[8]]
[1]  9 10

[[9]]
[1] 1 4 8

[[10]]
[1] 2 5 8

> edge2adj("Edgelist03.txt")
[[1]]
[1] 2 3

[[2]]
[1] 1 3 4 5

[[3]]
[1] 1 2 4 5

[[4]]
[1] 2 3 5 6

[[5]]
[1] 2 3 4 6
```

```
[[6]]
[1]  4  5  8 11 12

[[7]]
[1] 8

[[8]]
[1]  6  7  9 10

[[9]]
[1]  8 10

[[10]]
[1] 8 9

[[11]]
[1] 6

[[12]]
[1] 6
```