# Assignment 2 - Programming Part

Data Structures - Summer School

May 31, 2007

- *Deadline* - 23:59 hrs - June 3, 2007.
- *How to Submit*

  (1) For IITK students - You have to upload the final zip file at the ftp server
  ftp://172.26.77.168
  userid : sumschool
  passwd: sumschool
  Note: The server will be up around 21:00 hrs on Sunday (June 3, 2007)

  (2) For non IITK Students - Mail your final .zip file as the attachment to the mail id sumschool@cse.iitk.ac.in

- *File Format*

  Only .zip file will be accepted. No .rar/ .tar.gz/ .tgz / .java/ .C files please.
  The zip file should have the filename rollnumber-asn2.zip for example if your roll number is Y3214 then file name will be Y3214-asn2.zip. For non-iit students file name should be named as yourname-asn2.zip.

  *How to do all this ?*

  Create a folder/directory with name rollnumber-asn2 or yourname-asn2 (if you are non-iitk student). Now put your all codes appropriately in it. To zip the file - Windows users please use the software WinZip to create the rollnumber-asn2.zip or yourname-asn2.zip file. Linux users can use the command line

  prompt> zip -r rollnumber-asn2.zip rollnumber-asn2/

  For unzipping

  prompt> unzip rollnumber-asn2.zip

---

- **Problem Statement**

  Consider a graph $G$ with vertex set $V$ and edge set $E$. The graph has to be created dynamically in the following manner

  Let the graph has two nodes (say V1 and V2) and a single edge connecting them in the beginning. Now preferentially select a vertex among the existing vertices of the graph based on its degree i.e a vertex with higher degree should have higher chances of being selected. (So currently as both V1 and V2 have only a single edge and hence the degree=1, they are equally likely for getting selected). Introduce a new node (say V3) and connect it to the selected vertex in the previous step. Again preferentially select a vertex based on its degree in this new graph and connect it to a new node (say V4). Repeat this process till the number of nodes in the graph equals $N$. Hence for the graph G, number of vertices $= |V| = N$ and number of edges $= |E| = N-1$. Note $10^2 < N < 10^5$

**Hint** Preferential selection of vertex based on its degree means that if the degree of any vertex $V_i$ is $k_i$ then its selection probability equals $\frac{k_i}{\sum k_i}$. To achieve this one can use the following code

```
Object vertexToBeSelected = new Object();
double probDegree = rand.nextDouble()*(2*(graph.numberOfEdges()));
double sumDegree = 0;
for(int k=0; k < graph.numberOfVertices; k++)
{ sumDegree += (graph.degreeOf(vertices[k]));
if(sumDegree > probDegree)
{ vertexToBeSelected = vertices[k];
break; }
}
```

The idea is to first select a random number between 0 and 1 and multiply it by twice the current number of edges of the graph as $\sum k_i = 2 * currentNumberOfEdges$. Then iterate over the vertices of the graph summing over their degree till you reach the value of probDegree. You can imagine this like a fortune wheel or a pie chart with each vertex occupying a sector with area proportional to its degree. Selection then means rotating the wheel and looking for where the arrow head rests finally.

Thus to create the graph using the above approach you have to design appropriate data structure for the graph and have to implement various methods like numberOfEdges(), numberOfVertices(), degreeOf() - **(50 marks till here)**

After creating the graph, you have to plot its degree distribution i.e a x-y plot with y axis having $N(k)$ = number of nodes with degree k and x axis having $k$ = degree varying from 0 to maximum degree of the graph.

**Hint** To achieve this one can use the following code

```
DegreeDistribution[] = new int[graph.maxDegree()]
for(int k = 0;k<N;k++)
{ DegreeDistribution[graph.degreeOf(vertices[k])]++;
}
```

The idea is to first create an integer array of size equal to maximum degree of the graph. And then iterate over all the vertices, updating the count in the array corresponding to their degree. Thus DegreeDistribution[k] will represent number of nodes with degree $k$.

You have to print this degree distribution in a file named "distribution.txt". As we have to use this text file with a tool/software called gnuplot to plot the final x-y plot, you have to print it in a specific format only governed by this code **(75 marks till here)**

```
for(int k = 1;k<graph.maxDegree();k++)
{ System.out.println(k+" "+DegreeDistribution[k]);
}
```

To ensure that you have proceeded correctly till now, you can actually plot the degree distribution using the tool gnuplot (Available free for both windows and linux). You just need to type following commands on the gnuplot prompt.

```
gnuplot>set logscale xy
gnuplot>plot "distribution.txt"
```

Don't forget to set the directory path first. If everything is correct, you will get a linear plot. Why ? Think over this. Note that in above example, we have plotted $log(N(k))$ Vs $log(k)$ and thus linear nature implies that $N(k) \propto k^{-\gamma}$ where $\gamma$ is constant. **(25 marks for correct plot)**