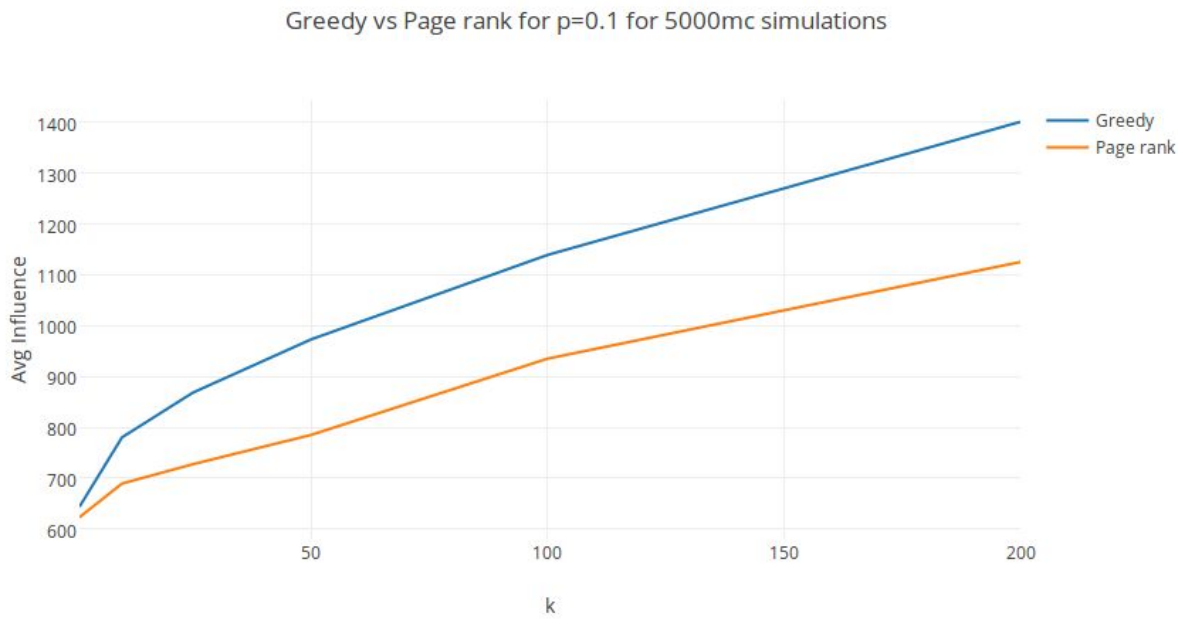
# Assignment Report: Influence Maximization Under IC Model

**Ques1.** Edge weight =0.1 with 5000 MC simulations, plot for the **expected spread** obtained with different k for PageRank and Greedy:

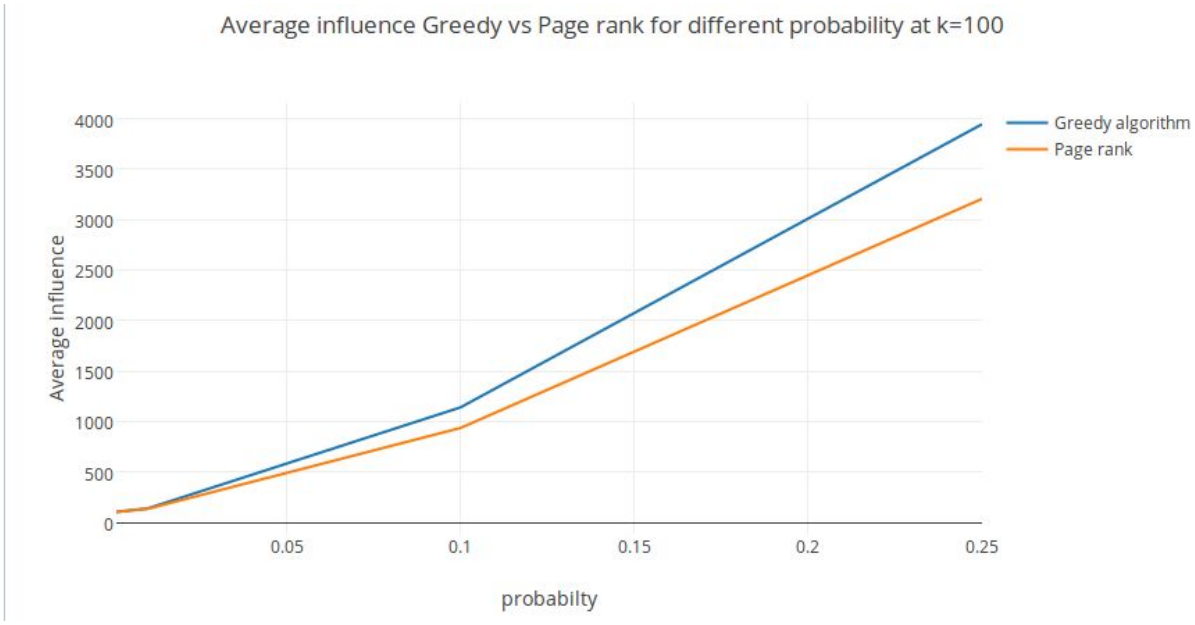



Corresponding readings:

| k | Greedy | Page rank |
|---|---|---|
| 1 | 644.008 | 623.319 |
| 10 | 780.4 | 689.534 |
| 25 | 868.384 | 727.606 |
| 50 | 973.009 | 785.177 |
| 100 | 1139.04 | 935.05 |
| 200 | 1400.73 | 1125.12 |

As we can see, the expected influence using greedy with probability=0.1 is always more than expected influence using page rank. This difference increases with increase in k. The reason behind this is that the **greedy algorithm selects the next node which gives highest increase in influence** after selecting a previous seed node whereas in page rank nodes are selected using the highest page rank. Two nodes may have same high page rank but those two nodes may be influencing same set of nodes, because of which the gain in influence of selecting one node after other will be zero. But this node with zero gain in influence will be selected as next node in case of page rank algorithm as it has high page rank. So, **page rank algorithm selects nodes which may be influencing same set of nodes(or almost same set of nodes)** and thus these nodes may be not give much increase in influence which is not in case of greedy algorithm. Hence greedy algorithm always have better influence than page rank algorithm as it always selects the node with highest gain in influence.

**Ques2.** Plot for the expected spread obtained using PageRank and Greedy at different probabilities and k=100:




Corresponding readings:

| Probability | Greedy algorithm | Page rank |
|---|---|---|
| 0.001 | 103.405 | 103.221 |
| 0.01 | 137.052 | 133.398 |
| 0.1 | 1139.04 | 935.05 |
| 0.25 | 3945.01 | 3205.44 |
| | | |

We can see, that the page rank gives almost same results for low probability wheras **as probabilities increases the influence from greedy algorithm are more than that of page rank algorithm.** The reason behind this is that a**t low probability,** the graph obtained will be very sparse i.e. each node is connected to very less nodes(in terms of probability). So two nodes with common nodes connected to them will be very less. So **two nodes with high page rank are most probable to be connected to very few common nodes.** Thus a high page rank node will also be the node with which we get high gain in influence. But **as probability increases** the graph becomes more dense(in terms of probability) and hence **greedy will now select nodes which gives highest increase in influence whereas page rank will select nodes with high page rank but influencing same set of nodes**(and hence nodes with low increase in influence). Thus greedy gives better result in increase in probability compared to page rank algoritm.

**Ques3.** In case of greedy algorithm we find expected influence of each node and select the node with highest gain influence. Calculating expected influence of each node takes O(n) time where n is the total no. of nodes in graph. This influence is calculated mc times, where mc is the total number of mc simulations done. Therefore the worst case time now is O(mc*n). This procedure is repeated for

each of the n nodes. Therefore the time now is O(n*mc*n) i.e O(mc*$n^2$). This is done each time we select a seed node. Thus for selecting k seed node the worst case time is:

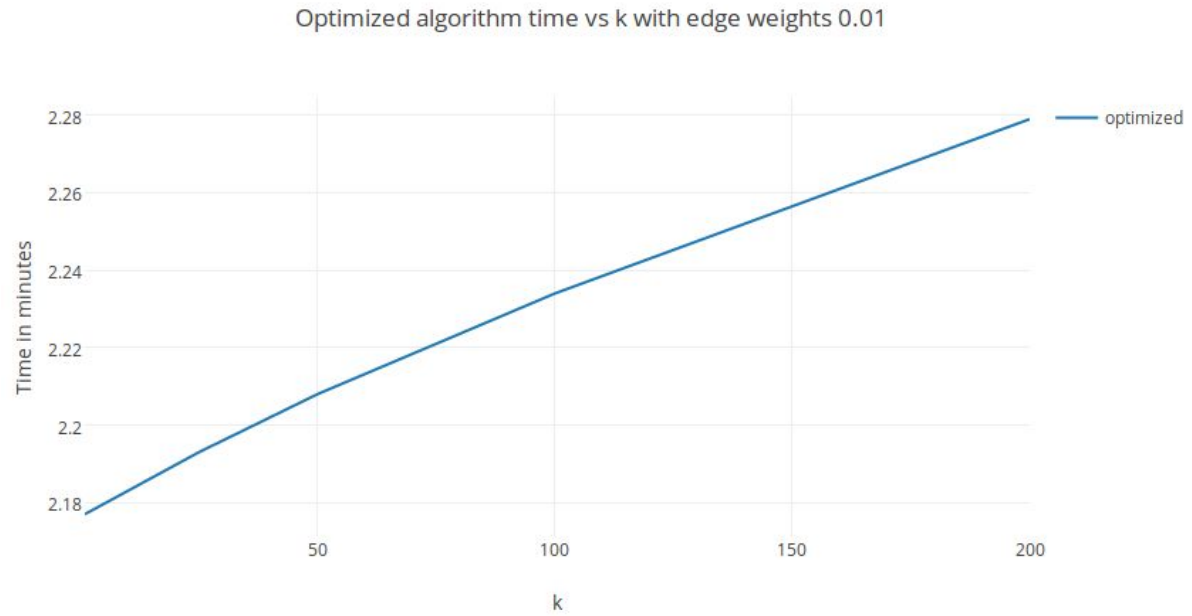
$$\mathbf{O(k*mc*n^2)}$$

**k=number of seed nodes**

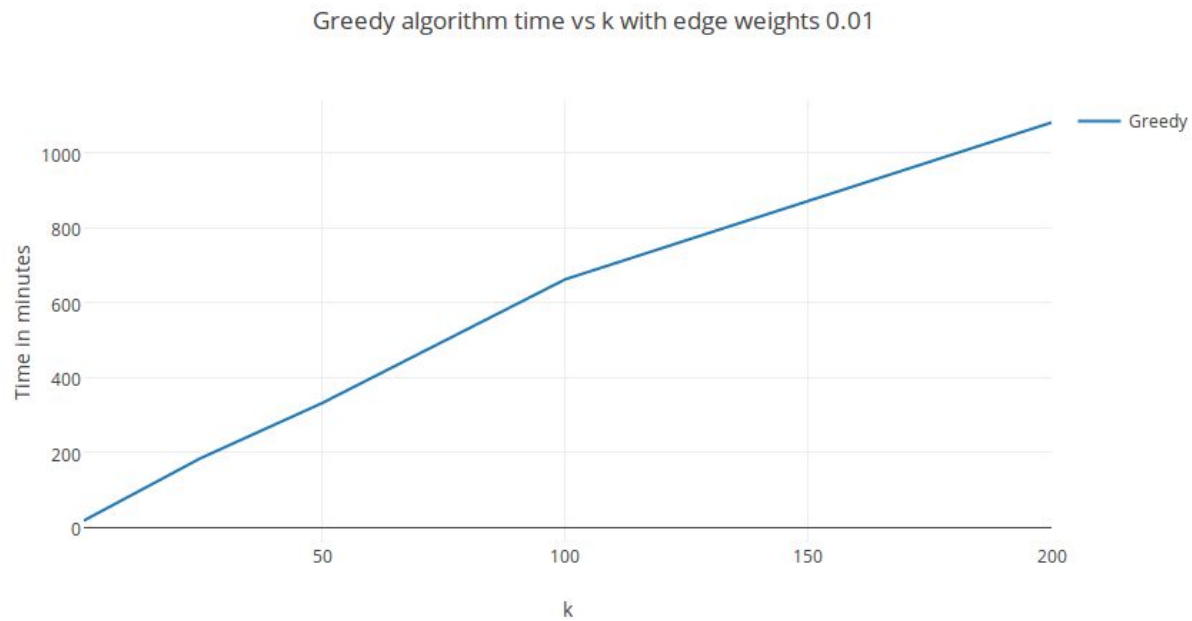
**mc=no of mc simulations performed**

**n=number of vertices in graph**

**Ques4a.** Plot for time taken in optimized algorithm for different values of k with edge weights 0.01. The time is in minutes:




Plot for time taken in greedy algorithm for different values of k with edge weights 0.01. The time is in minutes:

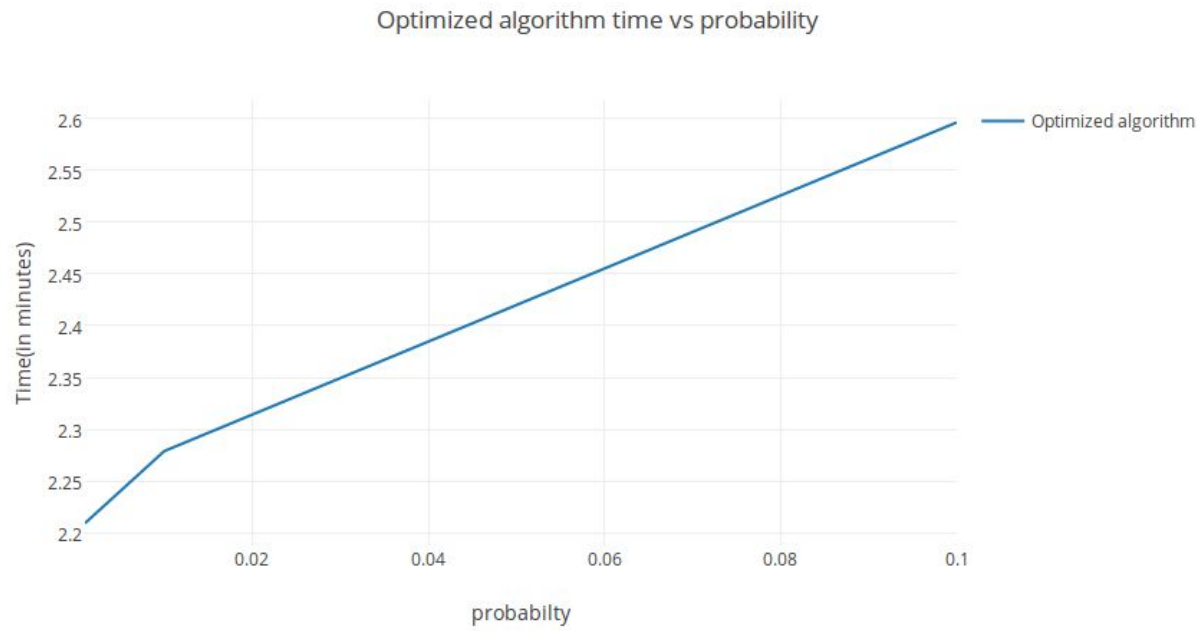


Combined plot for greedy vs optimized algorithm with edge weights 0.01 for different values of k:

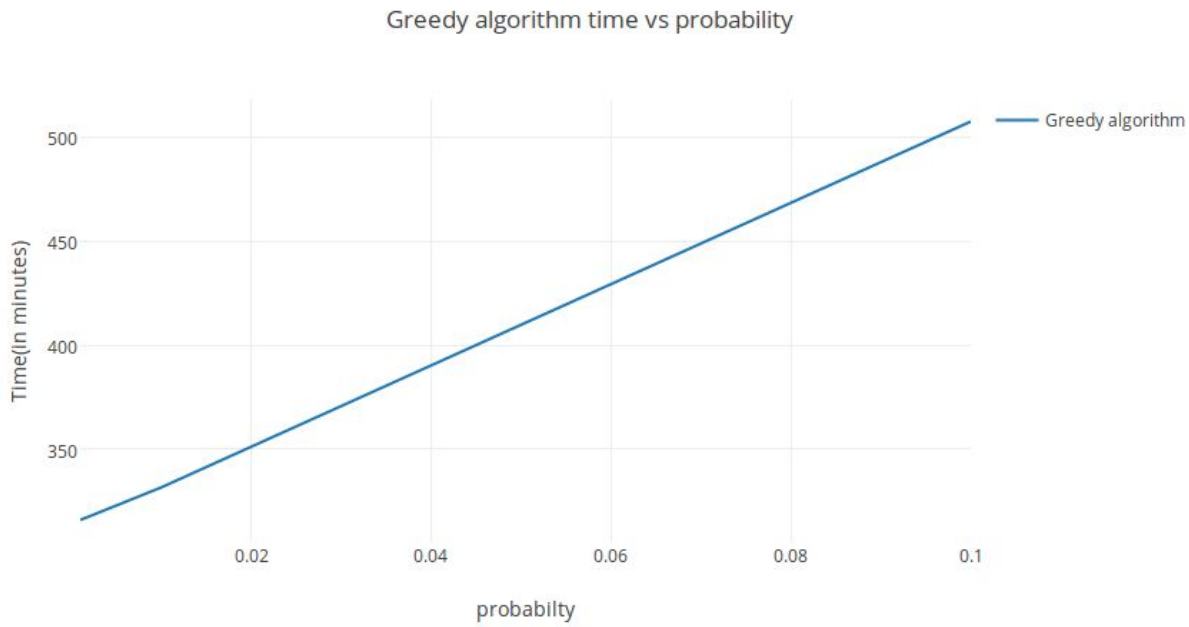

Corresponding readings for plot:

| k | optimized | Greedy |
|---|---|---|
| 1 | 2.177 | 18.03 |
| 10 | 2.183 | 80.52 |
| 25 | 2.193 | 184.48 |
| 50 | 2.208 | 331.63 |
| 100 | 2.234 | 662.01 |
| 200 | 2.279 | 1080.55 |

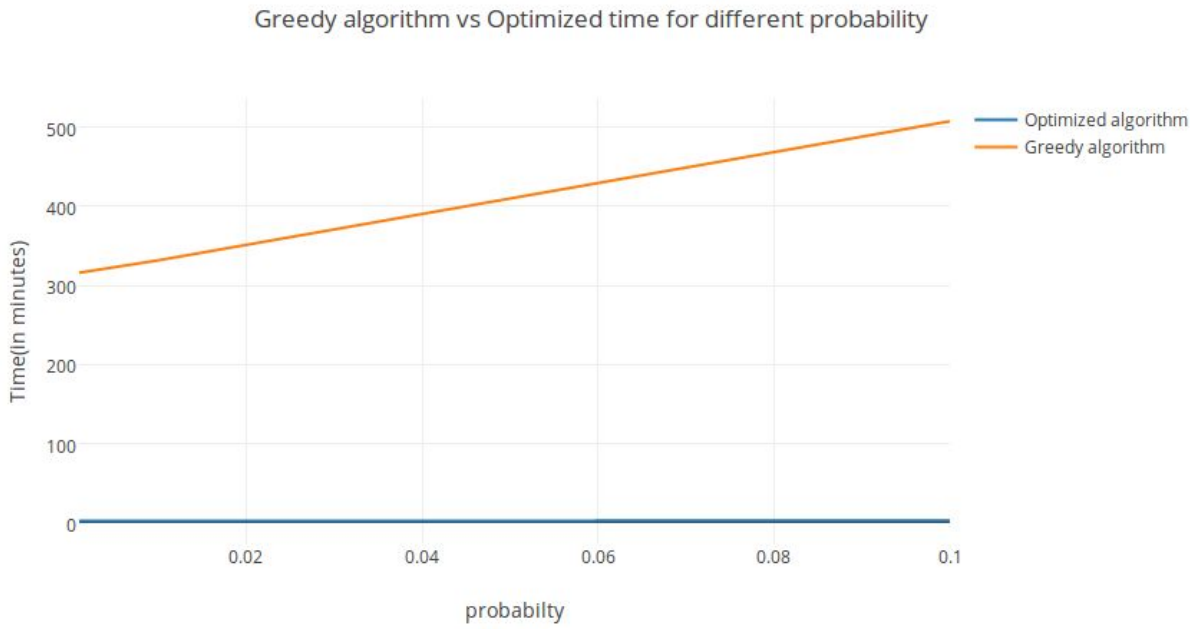
**Ques4.b.** Plot for time taken in optimized algorithm for different probabilities at k=50. The time is in minutes:




Plot for time taken in greedy algorithm for different probabilities at k=50. The time is in minutes:




Combined plot for greedy vs optimized algorithm with k=50 for different probabilities:




Corresponding readings for plot:

| probaility | Optimized algorithm | Greedy |
|---|---|---|
| 0.001 | 2.2095 | 315.95 |
| 0.01 | 2.208 | 331.63 |
| 0.1 | 2.596 | 507.47 |

The following paper has been cited for optimized algorithm:

- Improved Algorithms OF CELF and CELF++ for Influence Maximization
  By Jiaguo Lv, Jingfeng Guo , Zhen Yang, Wei Zhang and Allen Jocshi

The algorithm that has been implemented is **CELF algorithm.**

In this paper **the property of submodularity is used** to develop the optimized algorithm. The idea behind CELF is that **the marginal gain provided by a node in the current iteration cannot be better than the marginal gain provided by the node in the previous iteration**. The algorithm works as follows:

It maintains a max heap H that is sorted by marginal gain of nodes. In heap, each node corresponds to a node in the network.

In the first iteration, the marginal gain of every node is computed and the record is added to heap H in decreasing order of marginal gain of node. Then, in each iteration, for the first node in heap H, the algorithm will examine if marginal gain of top most node is last computed in the current iteration. If yes, due to the sub-modularity, top most node must be the node with the greatest marginal gain, and will be selected as the current seed. Otherwise, the marginal gain will be recomputed, and then the heap H with the new u.mg will be resorted. Obviously, through the optimization, the algorithm of CELF avoids the re-computation of marginal gain for each node in repeated iteration and thus provides same results in less time.