Submission Phase

1. Do assignment ☑ (/algorithmicthink1-003/human_grading/view/courses/975649/assessments/32/submissions)

Evaluation Phase

2. Evaluate peers ☑ (/algorithmicthink1-003/human_grading/view/courses/975649/assessments/32/peerGradingSets)

Results Phase

3. See results ☑ (/algorithmicthink1-003/human_grading/view/courses/975649/assessments/32/results/mine)

Your effective grade is 15

Your unadjusted grade is 15, which is simply the grade you received from your peers.

See below for details.

Overview

Graph exploration (that is, "visiting" the nodes and edges of a graph) is a powerful and necessary tool to elucidate properties of graphs and quantify statistics on them. For example, by exploring a graph, we can compute its degree distribution, pairwise distances among nodes, its connected components, and centrality measures of its nodes and edges. As we saw in the Homework and Project, breadth-first search can be used to compute the connected components of a graph.

In this Application, we will analyze the connectivity of a computer network as it undergoes a cyber-attack. In particular, we will simulate an attack on this network in which an increasing number of servers are disabled. In computational terms, we will model the network by an undirected graph and repeatedly delete nodes from this graph. We will then measure the *resilience* of the graph in terms of the size of the largest remaining connected component as a function of the number of nodes deleted.

Example graphs

In this Application, you will compute the resilience of several types of undirected graphs. We suggest that you begin by collecting and writing code to create the following three types of graphs:

- **An example computer network** - The text representation for the example network is here (http://storage.googleapis.com/codeskulptor-alg/alg_rf7.txt). You may use this provided code (http://www.codeskulptor.org/#alg_application2_provided.py) to load the file as an undirected graph (with

1239 nodes and 3047 edges). Note that this provided code also includes several useful helper functions that you should review.

- **ER graphs** - If you have not already implemented the pseudo-code for creating *undirected* ER graphs, you will need to implement this code. You may wish to modify your implementation of `make_complete_graph` from Project 1 to add edges randomly (again, keep in mind that in Project 1 the graphs were directed, and here we are dealing with undirected graphs).
- **UPA graphs** - In Application 1, you implemented pseudo-code that created DPA graphs. These graphs were directed (The D in DPA stands for directed). In this Application, you will modify this code to generate undirected UPA graphs. The UPA version should add an undirected edge to the UPA graph whenever you added a directed edge in the DPA algorithm. Note that since the degree of the newly added node is no longer zero, its chances of being selected in subsequent trials increases. In particular, you should either modify the `DPATrial class` to account for this change or use our provided UPATrial class (http://www.codeskulptor.org/#alg_upa_trial.py).

Important: Please use Coursera's "Attach a file" button to attach your plots/images for this Application as required. For each question you can attach more than one image as well as including text and math (LaTeX) in the same answer box. In particular, please do not host your solution plots/images on 3rd party sites. This practice exposes your peers to extra security risks and has the potential for abuse since the contents of a link to an external site may be modified after the hard deadline. Failure to follow this policy may lead to your plots/images being counted as "not submitted".

Question 1 (5 pts)

To begin our analysis, we will examine the resilience of the computer network under an attack in which servers are chosen at random. We will then compare the resilience of the network to the resilience of ER and UPA graphs of similar size.

To begin, you should determine the probability p such that the ER graph computed using this edge probability has approximately the same number of edges as the computer network. (Your choice for p should be consistent with considering each edge in the undirected graph exactly once, not twice.) Likewise, you should compute an integer m such that the number of edges in the UPA graph is close to the number of edges in the computer network. Remember that all three graphs being analyzed in this Application should have the same number of nodes and approximately the same number of edges.

Next, you should write a function `random_order` that takes a graph and returns a list of the nodes in the graph in some random order. Then, for each of the three graphs (computer network, ER, UPA), compute a random attack order using `random_order` and use this attack order in `compute_resilience` to compute the resilience of the graph.

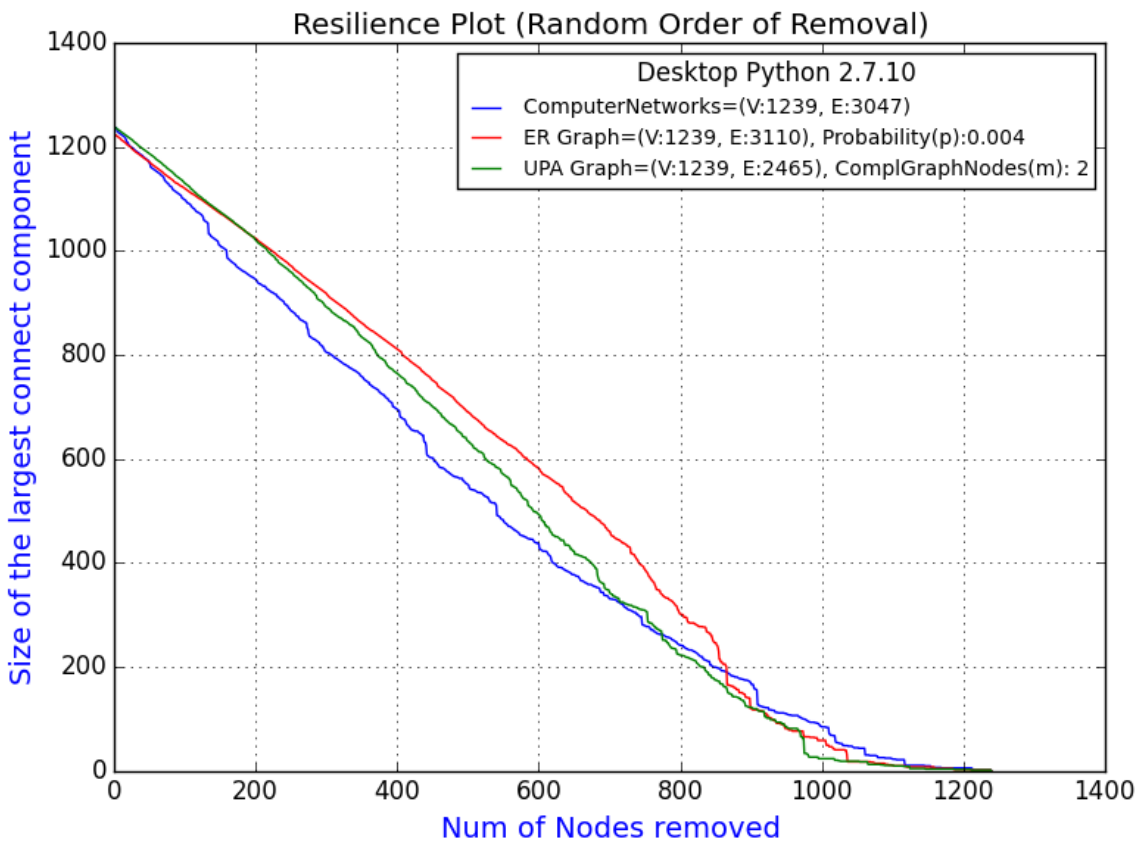
Once you have computed the resilience for all three graphs, plot the results as three curves combined in a single standard plot (not log/log). Use a line plot for each curve. The horizontal axis for your single plot be the the number of nodes **removed** (ranging from zero to the number of nodes in the graph) while the vertical axis should be the size of the largest connect component in the graphs resulting from the node removal. For this question (and others) involving multiple curves in a single plot, please include a legend in your plot that distinguishes the three curves. The text labels in this legend should include the values for p and m that you used in computing the ER and UPA graphs, respectively. Both `matplotlib` and `simpleplot` support these capabilities (matplotlib example (http://www.codeskulptor.org/#alg_legend_matplotlib.py) and simpleplot example (http://www.codeskulptor.org/#alg_legend_simpleplot.py)).

Note that three graphs in this problem are large enough that using CodeSkulptor to calculate `compute_resilience` for these graphs will take on the order of 3-5 minutes per graph. When using CodeSkulptor, we suggest that you compute resilience for each graph separately and save the results (or use

desktop Python for this part of the computation). You can then plot the result of all three calculations using `simpleplot`.

Once you are satisfied with your plot, upload your plot in the box below using "Attach a file" button (the button is disabled under the 'html' edit mode; you must be under the 'Rich' edit mode for the button to be enabled). Your plot will be assessed based on the answers to the following three questions:

- Does the plot follow the formatting guidelines (https://class.coursera.org/algorithmicthink1-002/wiki/ides?page=plotting) for plots?
- Does the plot include a legend? Does this legend indicate the values for p and m used in ER and UPA, respectively?
- Do the three curves in the plot have the correct shapes?



Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Item a (1 pt) Does the plot follow the formatting guidelines for plots?

The formatting guidelines include the following items:

- The plot is an image and not a text file.
- The plot is appropriately trimmed. Showing the boundary of the plot's window is fine. However, the plot should not include part of the desktop.
- The elements of the plot are of the correct type. Line plots are not the same as point

plots.

- Both axes should have tick marks labeled by regularly-spaced coordinate values.
- Both axes have appropriate text labels that describe the quantities being plotted.
- The plot has an appropriate title that describes the content of the plot.

Assess the submitted plot based on these guidelines.

As a specific note, if the horizontal axis is labeled "Number of nodes remaining" (as opposed to "Number of nodes removed" as specified in the Question), please score this item as a zero. This situation will be apparent if the three curves are increasing as opposed to decreasing. You will then grade these curves for correctness using the "flipped" solution given in item c.

Score from your peers: **1**

Item b (1 pt) Does the plot include a legend? Does this legend include correct values for p and m used in ER and UPA, respectively?

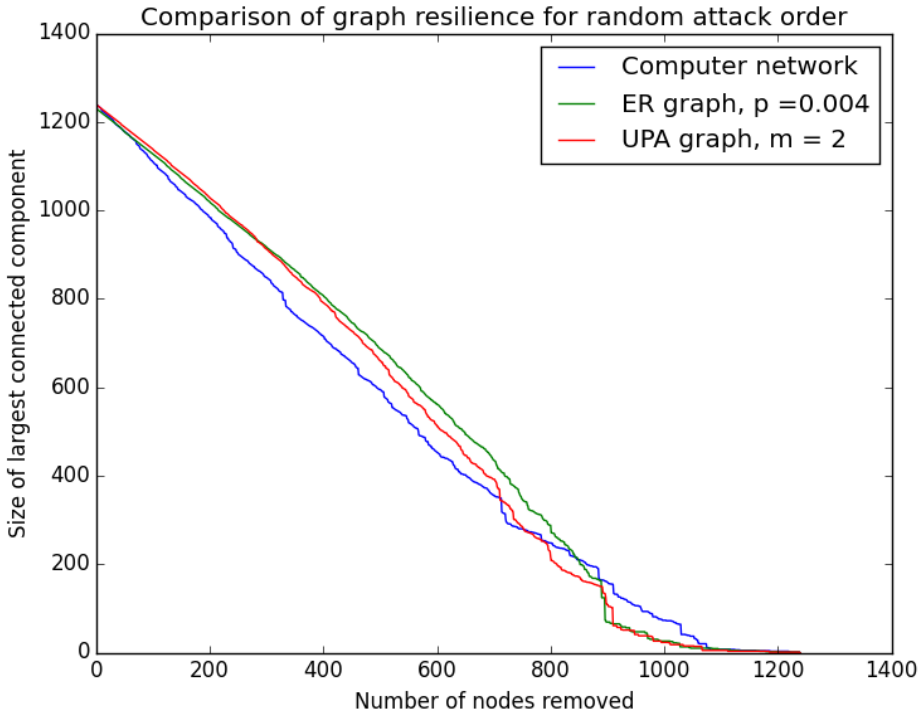
The correct value of p is approximately 0.004. Score values for p between 0.0039 and 0.0040 (inclusive) as being correct. The correct value for m is either 2 or 3.

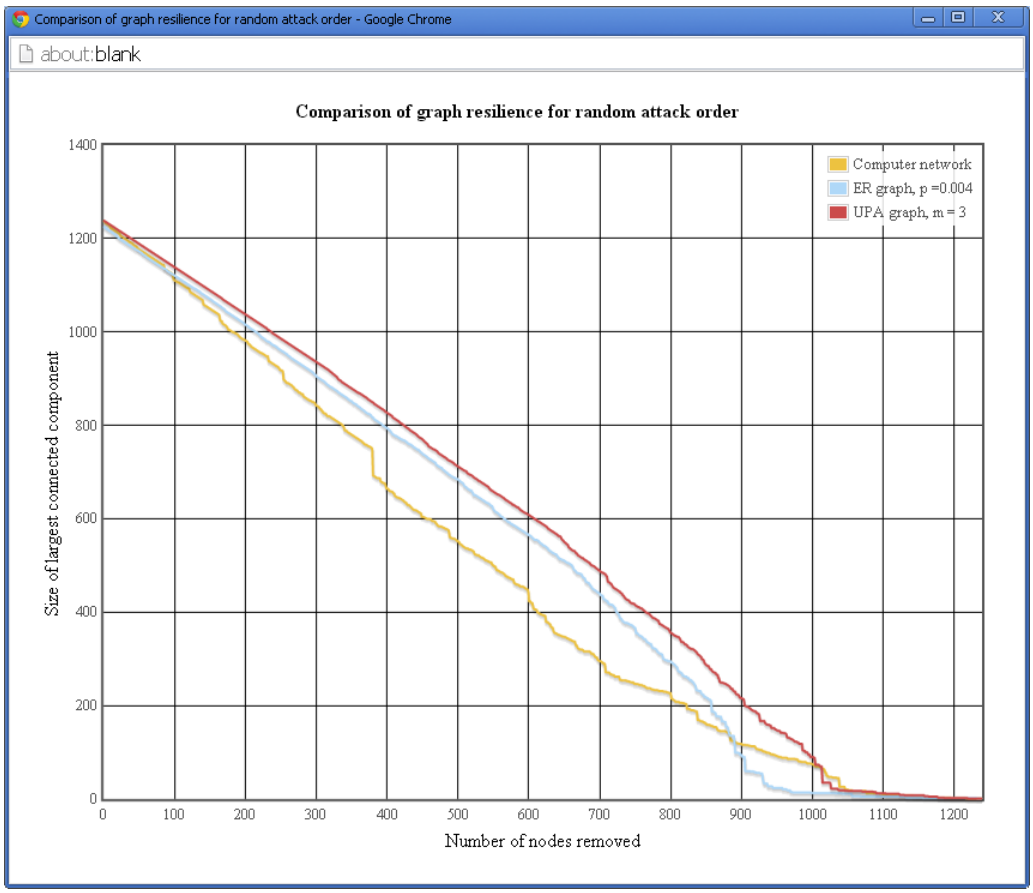
Some students' ER methods may consider each possible edge twice. As a consequence, they may have submitted a value $p \approx 0.002$. Considering each edge twice is inconsistent with the supplied pseudo-code for creating an undirected ER graph. (Note that there is a hint in the problem warning against this choice.) Therefore, count this choice for p as being incorrect.

Score from your peers: **1**

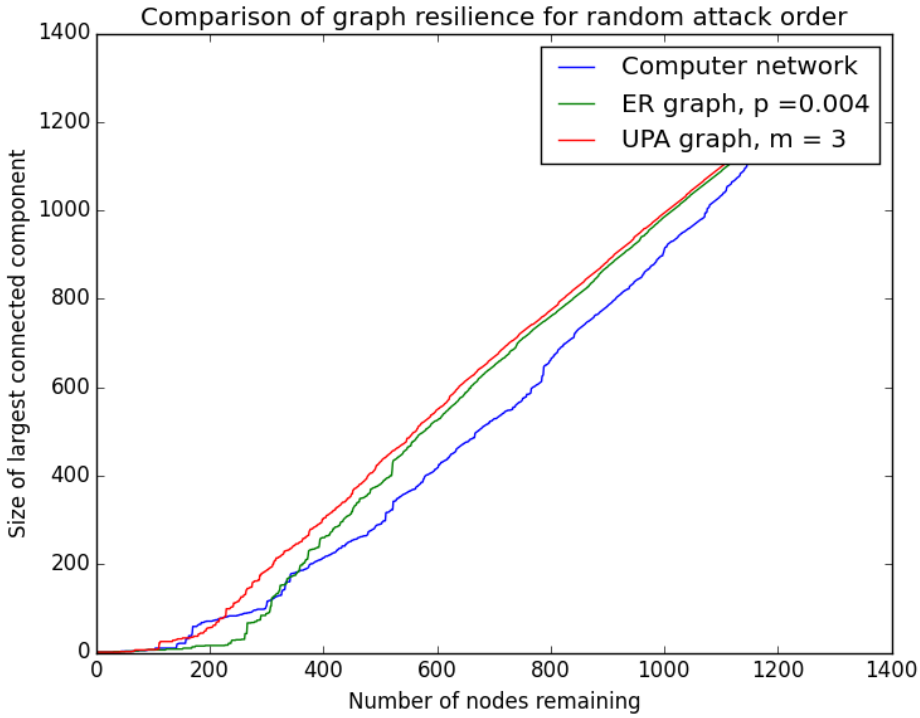
Item c (3 pts) Do the three curves in the plot have the correct shapes?

Below are correct plots for $m = 2$ in `matplotlib` and $m = 3$ in `simpleplot`. All three curves should descend in a roughly linear manner towards zero as the number of nodes removed increases. Note that due to the use of random attack orders, the three curves may differ somewhat from the plots below. However, you should see a downward linear trend with possible occasional sudden dips in all three curves.





This plot shows a `matplotlib` plot for $m = 3$ with the horizontal axis incorrectly varying by the number of nodes remaining. In this case, three curves increasing in a roughly linear manner. Since you should have deducted for this error in item a, please score plots that match this answer with full credit on this item.



Note that, if the plots used incorrect values for either p or m, judge the correctness of the submitted plots based on their similarity to the shape of the correct plots. Finally, deduct one point from the score below if three separate (correct) plots were submitted.

Score from your peers: **3**

Comments: Please enter an explanation for your scoring, especially if you deducted any points for one of the rubric items for this question.

peer 1 → *[This area was left blank by the evaluator.]*

peer 2 → good job

peer 3 → *[This area was left blank by the evaluator.]*

peer 4 → *[This area was left blank by the evaluator.]*

peer 5 → *[This area was left blank by the evaluator.]*

Question 2 (1 pt)

Consider removing a significant fraction of the nodes in each graph using `random_order`. We will say that a graph is *resilient* under this type of attack if the size of its largest connected component is roughly (within ~25%) equal to the number of nodes remaining, after the removal of each node during the attack.

Examine the shape of the three curves from your plot in Question 1. Which of the three graphs are resilient under random attacks as the first 20% of their nodes are removed? Note that there is no need to compare the three curves against each other in your answer to this question.

Computer Networks Graph: On 20% node removal in random order, the largest connected component is > 85% of total nodes. Thus resilient
EA Graph: On 20% node removal in random order, the largest connected component is ~ 75% of total nodes. On successive multiple runs it was around 72% in couple of runs. But can be called resilient.
UPA Graph: On 20% node removal in random order, the largest connected component is > 80% of total nodes. Thus resilient

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Item a (1 pt) Which of the three graphs are resilient under random attack as the first 20% of their nodes are removed?

All three graphs are resilient under random attack as the first 20% of their nodes are removed (unless the random order happens to remove a large number of high degree nodes, which is unlikely). In particular, all three curves have an approximate slope of -1 as the number of nodes removed varies from 0 to 240.

If the submitted answer states that one of the graphs is not resilient to a random attack, check the corresponding resilience curve in Question 1. If there is a large drop (> 25%) in the resilience for this curve when the first 20% of the nodes are removed, assume that the submitter was unlucky and count the submitted answer for this graph as being correct. (If you have doubt about whether the drop is 25%, give your peer the benefit of the doubt.) Otherwise, count the answer as being incorrect.

Score from your peers: **1**

Comments: Please enter an explanation for your scoring, especially if you deducted any points for one of the rubric items for this question.

peer 1 → *[This area was left blank by the evaluator.]*

peer 2 → good

peer 3 → *[This area was left blank by the evaluator.]*

peer 4 → *[This area was left blank by the evaluator.]*

peer 5 → *[This area was left blank by the evaluator.]*

Question 3 (3 pts)

In the next three problems, we will consider attack orders in which the nodes being removed are chosen based on the structure of the graph. A simple rule for these *targeted attacks* is to always remove a node of maximum (highest) degree from the graph. The function `targeted_order(ugraph)` in the provided code (http://www.codeskulptor.org/#alg_application2_provided.py) takes an undirected graph `ugraph` and iteratively does the following:

- Computes a node of the maximum degree in `ugraph`. If multiple nodes have the maximum degree, it chooses any of them (arbitrarily).
- Removes that node (and its incident edges) from `ugraph`.

Observe that `targeted_order` continuously updates `ugraph` and always computes a node of maximum degree with respect to this updated graph. The output of `targeted_order` is a sequence of nodes that can be used as input to `compute_resilience`.

As you examine the code for `targeted_order`, you feel that the provided implementation of `targeted_order` is not as efficient as possible. In particular, much work is being repeated during the location of nodes with the maximum degree. In this question, we will consider an alternative method (which we will refer to as `fast_targeted_order`) for computing the same targeted attack order. Here is a pseudo-code description of the method:

Algorithm 1: FastTargetedOrder.

Input: Graph $g = (V, E)$, with $V = \{0, 1, \ldots, n-1\}$.
Output: A (ordered) list L of the nodes in V in decreasing order of their degrees.

for $k \leftarrow 0$ *to* $n-1$ **do**
  $DegreeSets[k] \leftarrow \emptyset$; // $DegreeSets[k]$ is a set of all nodes whose degree is k
for $i \leftarrow 0$ *to* $n-1$ **do**
  $d \leftarrow degree(i)$; // d is the degree of node i
  $DegreeSets[d] \leftarrow DegreeSets[d] \cup \{i\}$;
$L \leftarrow []$; // L is initialized to an empty list
$i \leftarrow 0$;
for $k \leftarrow n-1$ **downto** 0 **do**
  while $DegreeSets[k] \neq \emptyset$ **do**
    Let u be an arbitrary element in $DegreeSets[k]$;
    $DegreeSets[k] \leftarrow DegreeSets[k] - \{u\}$;
    foreach *neighbor* v *of* u **do**
      $d \leftarrow degree(v)$;
      $DegreeSets[d] \leftarrow DegreeSets[d] - \{v\}$;
      $DegreeSets[d-1] \leftarrow DegreeSets[d-1] \cup \{v\}$;
    $L[i] \leftarrow u$;
    $i \leftarrow i + 1$;
    Remove node u from g;
return L;

In Python, this method creates a list `degree_sets` whose kth element is the set of nodes of degree k. The method then iterates through the list `degree_sets` in order of decreasing degree. When it encounter a non-empty set, the nodes in this set must be of maximum degree. The method then repeatedly chooses a node from this set, deletes that node from the graph, and updates `degree_sets` appropriately.

For this question, your task is to implement `fast_targeted_order` and then analyze the running time of these two methods on UPA graphs of size n with $m = 5$. Your analysis should be both mathematical and empirical and include the following:

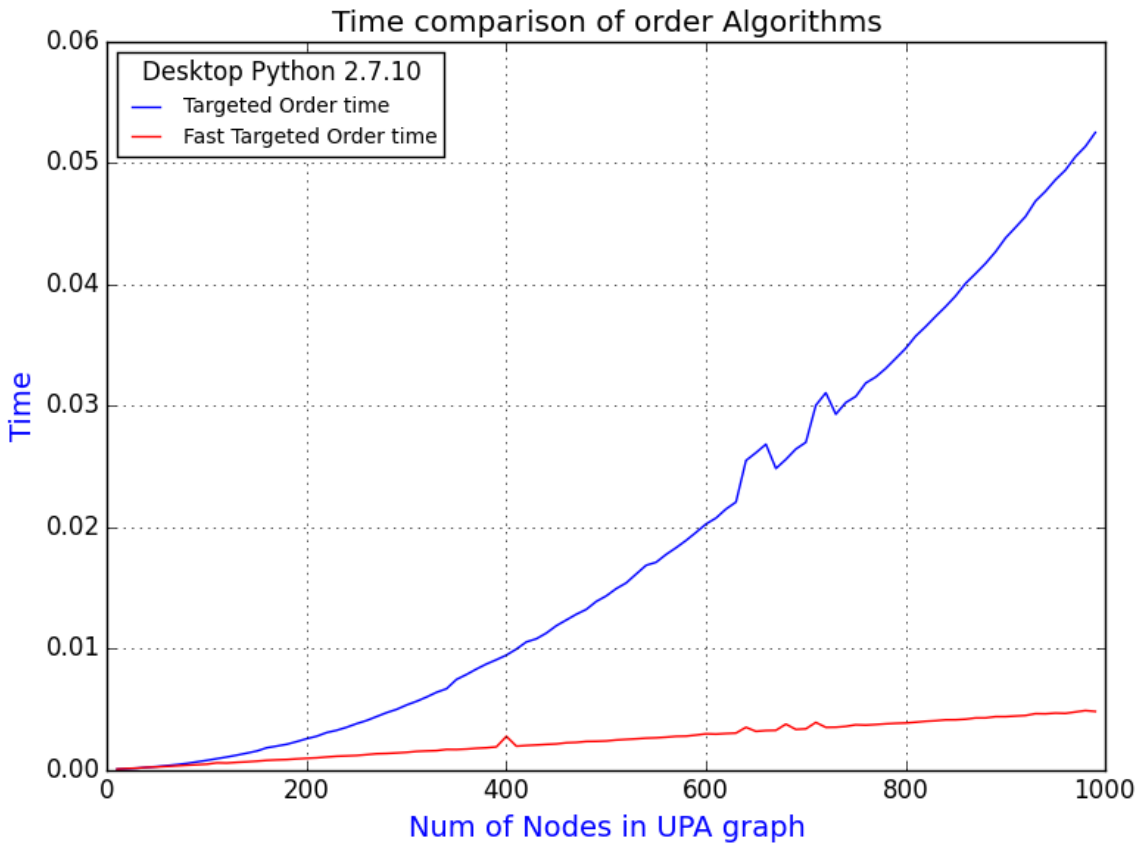
- Determine big-O bounds of the worst-case running times of `targeted_order` and `fast_targeted_order` as a function of the number of nodes n in the UPA graph.
- Compute a plot comparing the running times of these methods on UPA graphs of increasing size.

Since the number of edges in these UPA graphs is always less than $5n$ (due to the choice of $m = 5$), your big-O bounds for both functions should be expressions in n. You should also assume that the all of the set operations used in `fast_targeted_order` are $O(1)$.

Next, run these two functions on a sequence of UPA graphs with n in `range(10, 1000, 10)` and $m = 5$ and use the `time` module (or your favorite Python timing utility) to compute the running times of these functions. Then, plot these running times (vertical axis) as a function of the number of nodes n (horizontal axis) using a standard plot (not log/log). Your plot should consist of two curves showing the results of your timings. Remember to format your plot appropriately and include a legend. **The title of your plot should indicate the implementation of Python (desktop Python vs. CodeSkulptor) used to generate the timing results.**

Your answer to this question will be assessed according to the following three items:

- What are tight upper bounds on the worst-case running times of `targeted_order` and `fast_targeted_order`? Use big-O notation to express your answers (which should be very simple).
- Does the plot follow the formatting guidelines for plots? Does the plot include a legend? Does the title include the implementation of Python used to compute the timings?
- Are the shapes of the timing curves in the plot correct?



Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Item a (1 pt) What are tight upper bounds on the worst-case running times of

`targeted_order` and `fast_targeted_order`? Use big-O notation to express your answers (which should be very simple).

The running time for `targeted_order` is $O(n^2)$. Since the code is simple, we will leave the analysis to you.

Lines 1-5 in `fast_targeted_order` take $O(n+m)$ time. The running time for remaining portion of `fast_targeted_order` is also $O(n+m)$. To arrive at this bound, we note that `fast_targeted_order` performs a constant number of single item set operations for each node and edge in the graph. Since each of these set operations are assumed to $O(1)$, the running time for this algorithm is $O(n+m)$ where m is the number of edges. Since number of edges in the UPA graph is bounded by $5n$, the running time for all of `fast_targeted_order` is $O(n)$.

Note that the submitted answer does not need to provide this analysis. It only needs to provide the correct estimates using big-O notation. In the case where the submitted answer includes the number of edges m, you should score the running times of $O(n^2+m)$ for `targeted_order` as being correct and answer of the form $O(n+m)$ for `fast_targeted_order` as being correct.

If the submitted answers are different, feel free to use your math skills to determine if they are equivalent to the answers above. However, there is no need to be exhaustive here.

Score from your peers: **0**

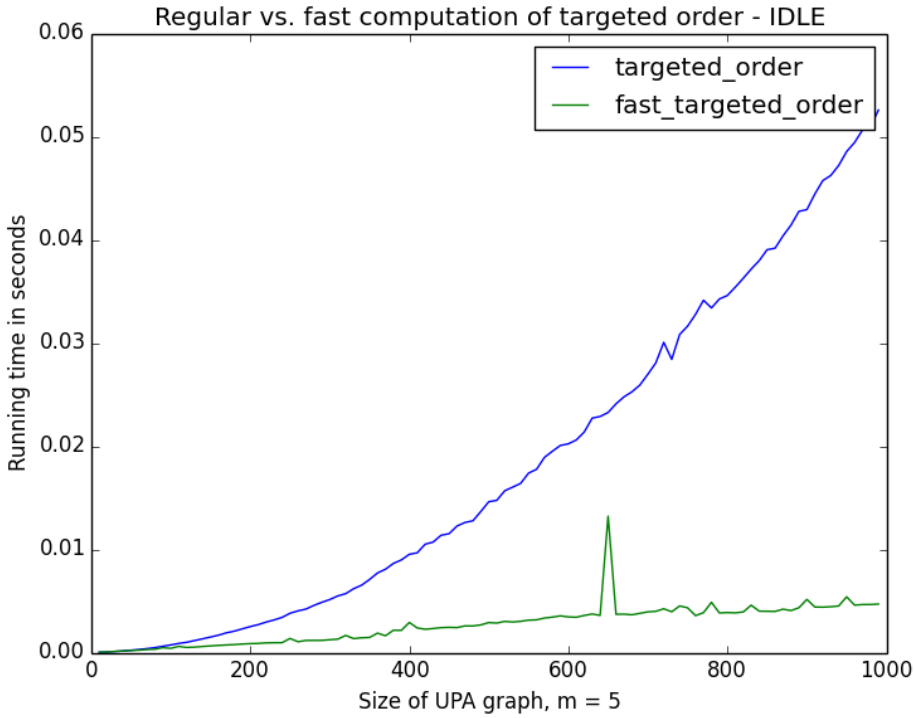
item b (1 pt) Does the plot follow the formatting guidelines for plots? Does the plot include a legend? Does the title include the implementation of Python used to compute the timings?

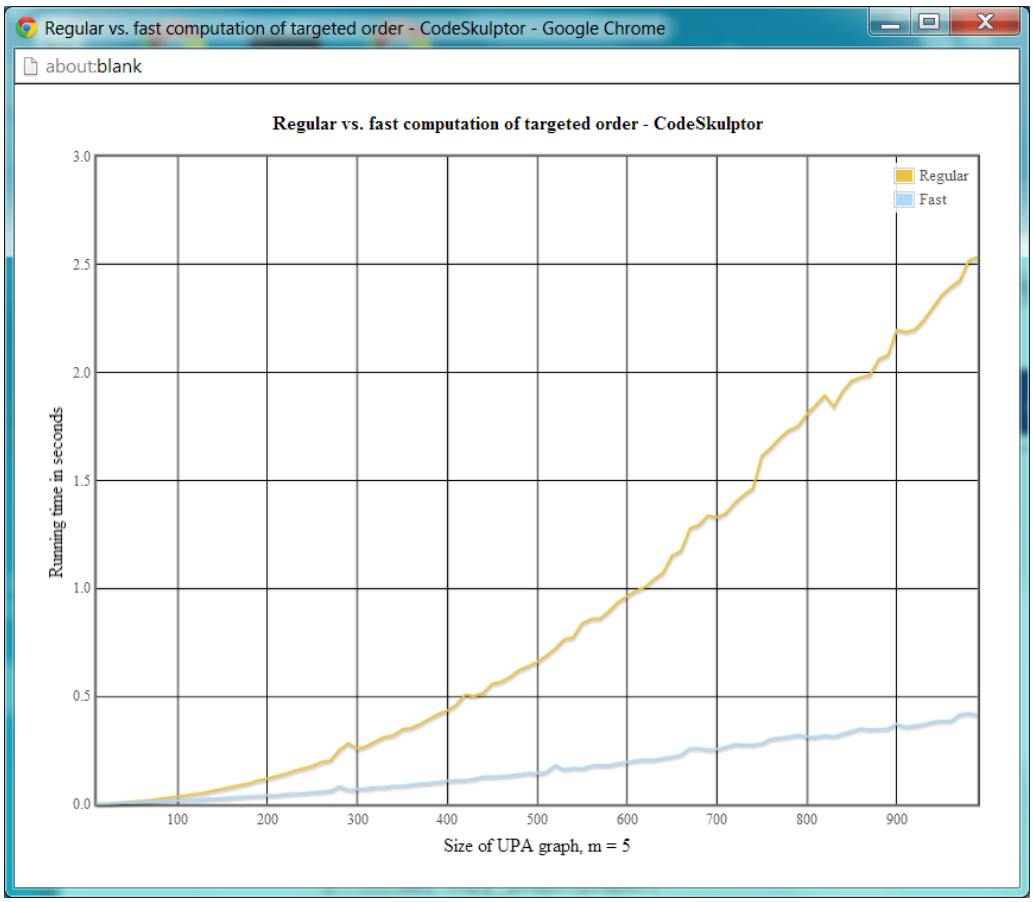
Reference the formatting guidelines from Question 1 item a for this item. The title of the plot should give some indication of what implementation of Python was used when running the program to help understand the vertical scale. The main point of this item is to distinguish desktop Python from CodeSkulptor due to the significant gap in their performance. Be generous here when grading and accept anything that sounds plausible. Answers like "desktop Python", "command line Python", "CPython", "IPython", and "CodeSkulptor" are fine. IDE names like "IDLE", "Anaconda" and "Canopy" are also fine. The user does not need to include version information.

Score from your peers: **1**

Item c (1 pt) Are the shapes of the timing curves in the plot correct?

Here are two examples of correct plots using an $O(n)$ implementation of lines 1-2: one using IDLE and `matplotlib` and another using CodeSkulptor and `simpleplot`.





Note that the timing curve for `fast_targeted_order` should be approximately linear while the timing curve for `targeted_order` should be quadratic. The scale on vertical axis may vary significantly since different machines have different processing power. For example, plots generated in CodeSkulptor may require 20-50 times more time than plots generated in IDLE.

Also, note that the curves in the plot may be somewhat noisy (bumpy). This shape is the result of Python's garbage collector (and other activities on the computer) slowing Python's performance and should not affect your scoring.

Score from your peers: **1**

Comments: Please enter an explanation for your scoring, especially if you deducted any points for one of the rubric items for this question.

peer 1 → *[This area was left blank by the evaluator.]*

peer 2 → good but needed the explicit answer about the order, not just graph

peer 3 → *[This area was left blank by the evaluator.]*

peer 4 → *[This area was left blank by the evaluator.]*

peer 5 → Seems like you forgot about running times in O-notation. But good job on what was submitted.

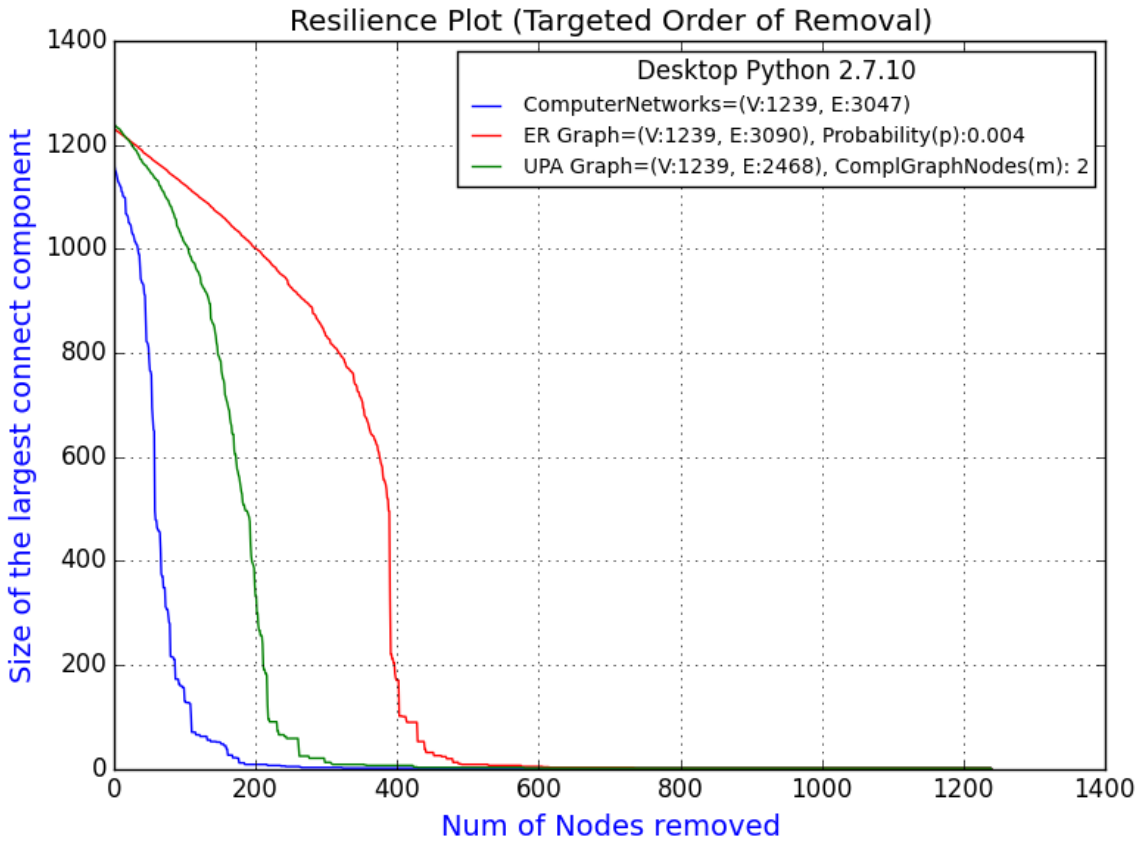
Question 4 (5 pts)

To continue our analysis of the computer network, we will examine its resilience under an attack in which servers are chosen based on their connectivity. We will again compare the resilience of the network to the resilience of ER and UPA graphs of similar size.

Using `targeted_order` (or `fast_targeted_order`), your task is to compute a targeted attack order for each of the three graphs (computer network, ER, UPA) from Question 1. Then, for each of these three graphs, compute the resilience of the graph using `compute_resilience`. Finally, plot the computed resiliences as three curves (line plots) in a single standard plot. As in Question 1, please include a legend in your plot that distinguishes the three plots. The text labels in this legend should include the values for p and m that you used in computing the ER and UPA graphs, respectively.

Once you are satisfied with your plot, upload your plot in the box below using "Attach a file" button (the button is disabled under the 'html' edit mode; you must be under the 'Rich' edit mode for the button to be enabled). Your plot will be assessed based on the answers to the following three questions:

- Does the plot follow the formatting guidelines for plots?
- Does the plot include a legend? Does this legend indicate the values for p and m used in ER and UPA, respectively?
- Do the three curves in the plot have the correct shape?



Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Item a (1 pt) Does the plot follow the formatting guidelines for plots?

Assess the submitted plot based using the same guidelines as provided in Question 1 item a.

Score from your peers: **1**

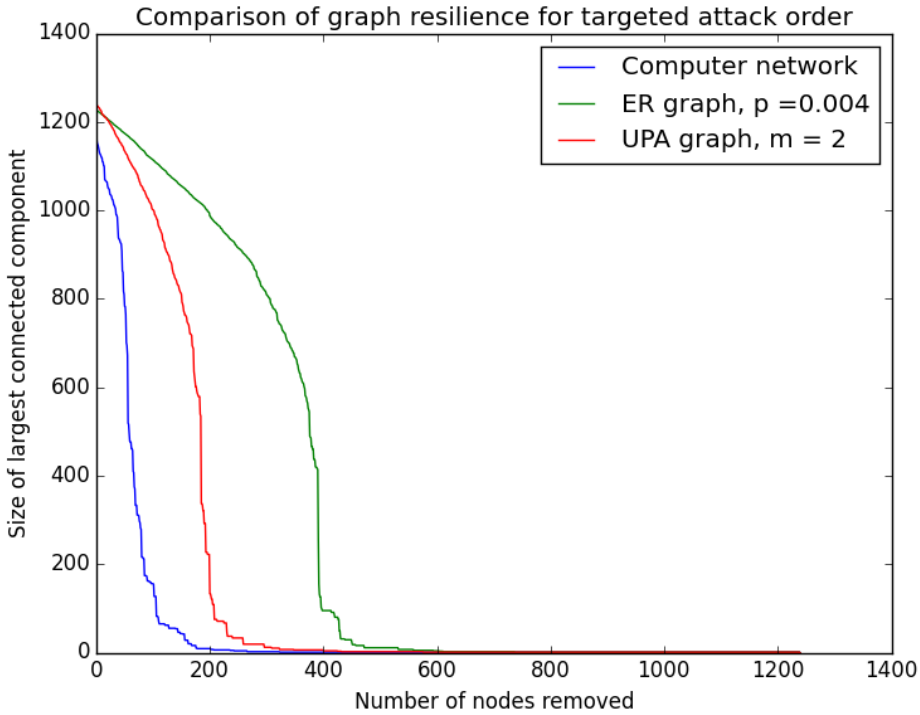
Item b (1 pt) Does the plot include a legend? Does this legend indicate the values for p and m used in ER and UPA, respectively?

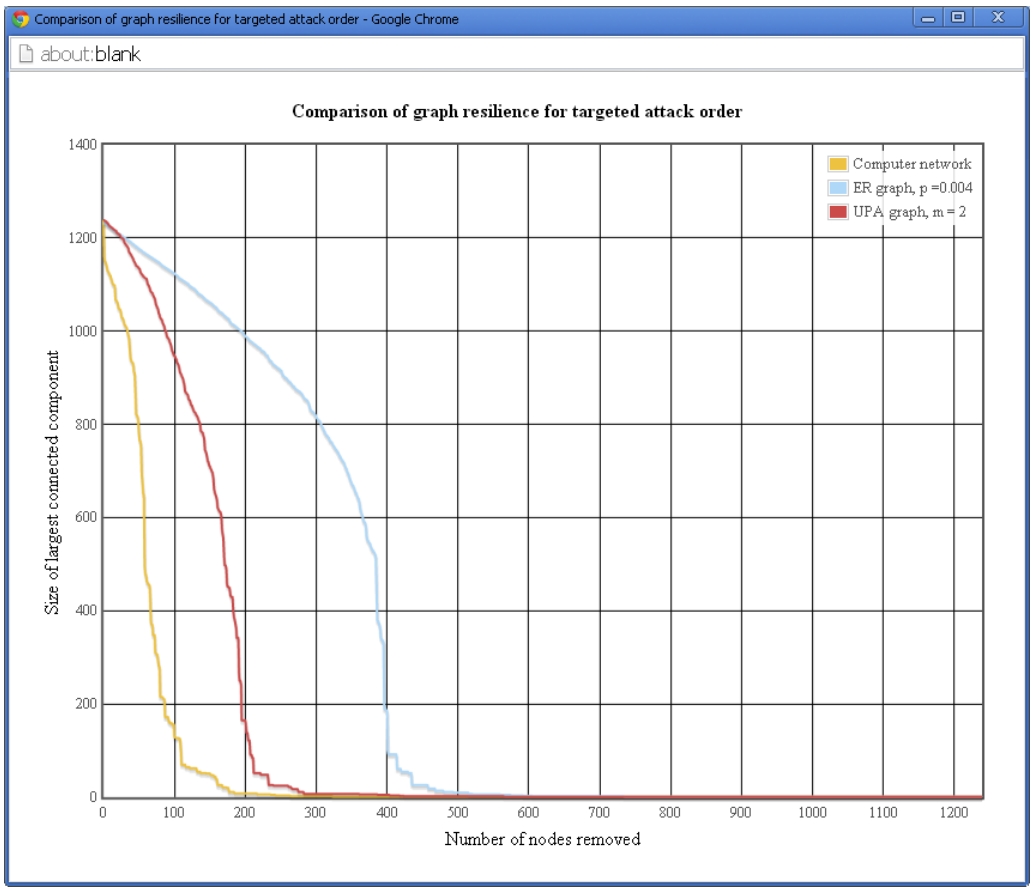
For this problem, you should **not** deduct a point if the provided legend does not contain values for p and m (or displays incorrect values). However, do score as a zero if no legend was provided.

Score from your peers: **1**

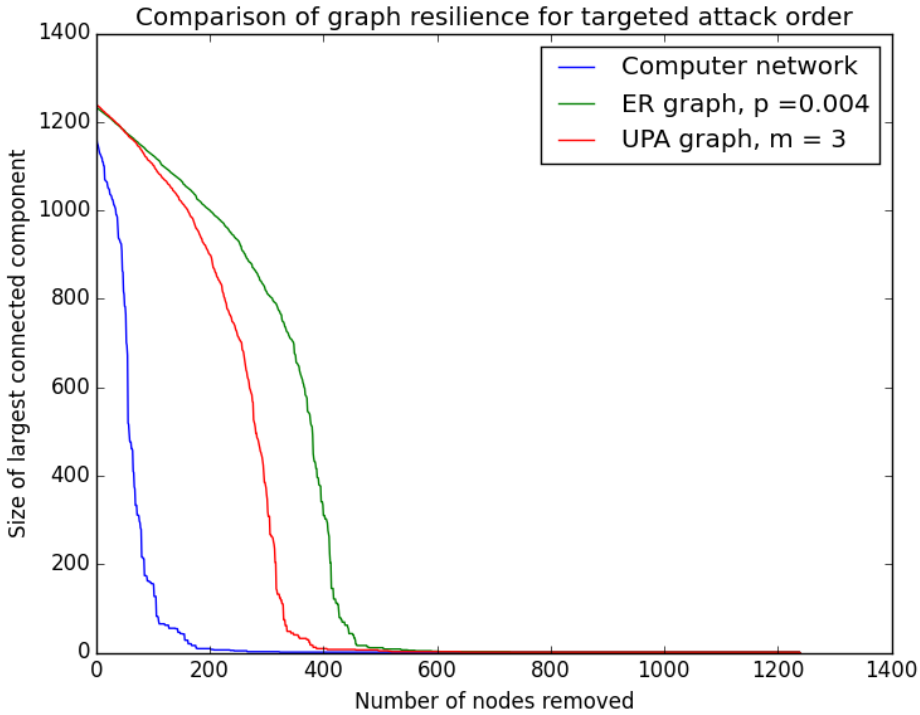
Item c (3 pts) Do the three curves in the plot have the correct shape?

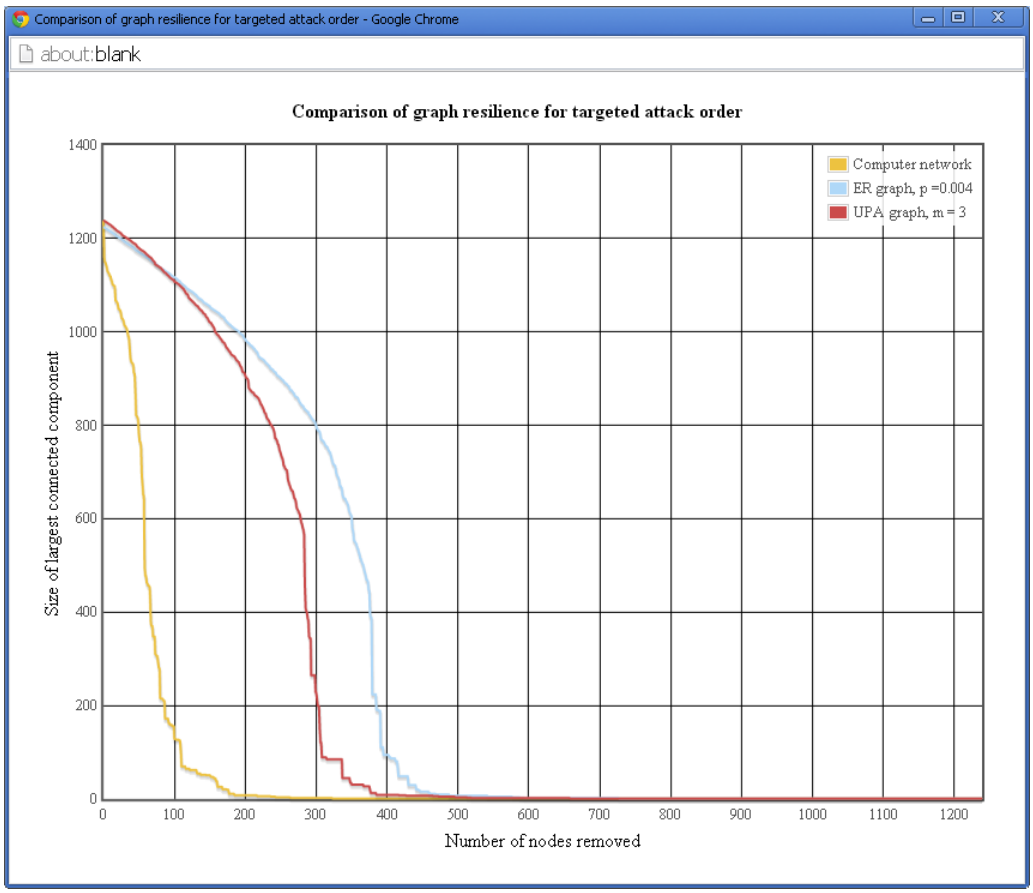
Below are `matplotlib` and `simpleplot` plots for $m = 2$.





Here are `matplotlib` and `simpleplot` plots for $m = 3$.





In general, the resilience for all three graphs should drop towards zero very quickly with the ER graph being most resilient and the computer network being least resilient. If the submitted plot for the $m = 3$ UPA curve is shifted significantly to the left and instead looks like the solution plot for the $m = 2$ UPA curve, score that curve as being incorrect since this shift indicates that incorrect probabilities were used in computing the UPA graph.

For this question, you should not deduct points for flipping the horizontal axis (i.e., using the label "Number of nodes remaining") or using three separate plots.

Score from your peers: **3**

Comments: Please enter an explanation for your scoring, especially if you deducted any points for one of the rubric items for this question.

peer 1 → *[This area was left blank by the evaluator.]*

peer 2 → good job

peer 3 → *[This area was left blank by the evaluator.]*

peer 4 → *[This area was left blank by the evaluator.]*

peer 5 → *[This area was left blank by the evaluator.]*

Question 5 (1 pt)

Now, consider removing a significant fraction of the nodes in each graph using `targeted_order`. Examine the shape of the three curves from your plot in Question 4. Which of the three graphs are resilient under targeted attacks as the first 20% of their nodes are removed? Again, note that there is no need to compare the three curves against each other in your answer to this question.

ER Graph is resilient.
Computer Network is NOT resilient: the size of largest connected component falls way below 25% when 20% of the nodes are removed in Targeted order.
UPA graph is NOT resilient: the size of largest connected component falls way below 25% when 20% of the nodes are removed in Targeted order.

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Item a (1 pt) Which of the three graphs are resilient under targeted attacks as the first 20% of their nodes are removed?

The slope of the ER curve is close to -1 as the first 20% of the nodes in the graph are removed. Therefore, the ER graph is resilient to targeted attacks of this type.

On the other hand, the size of the largest connected component in the computer network drops to almost zero when the first 20% of the nodes are removed. Therefore, the computer network should not be listed as being resilient.

The UPA graph is a close call. For $m = 2$, the size of the largest connected component drops to zero fairly quickly as 20% of the nodes are removed. For $m = 3$, this size is fairly close to the number of nodes remaining as 20% of the nodes are removed and is just

starting to drop signficantly. So, either choice (listing the UPA graph as being resilient or not listing the UPA graph as being resilient) should be counted as correct.

Score from your peers: **1**

Comments: Please enter an explanation for your scoring, especially if you deducted any points for one of the rubric items for this question.

peer 1 → *[This area was left blank by the evaluator.]*

peer 2 → good job

peer 3 → *[This area was left blank by the evaluator.]*

peer 4 → *[This area was left blank by the evaluator.]*

peer 5 → *[This area was left blank by the evaluator.]*

Question 6 (1 pt extra credit)

An increasing number of people with malevolent intent are interested in disrupting computer networks. If you found one of the two random graphs to be more resilient under targeted attacks than the computer network, do you think network designers should always ensure that networks' topologies follow that random model? Think about the considerations that one might have to take into account when designing networks and provide a short explanation for your answer.

During targeted order of node removal ER graphs perform better than UPA graphs. So it makes sense to use ER over UPA for the benefit of connection % during attack. But, all network topologies can not be satisfied using the EA graph. Consider following two cases:
- multiple network connection between two components/nodes so that to increase network bandwidth for data transfer.
- Some components/nodes should be connected mandatory for the system to work.
In these two scenarios and other creating a initial complete graph is necessary and thus DPA is useful. ER algorithm will not guarantee these network links between nodes and thus can not be used. When the network links are more generic like the ones in hubs/generic switches, using ER is beneficial over UPA.

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Item a (1 pt) If you found one of the two random graphs to be more resilient under targeted attacks than the computer network, do you think network designers should always

ensure that networks' topologies follow that random model?

No. Many factors, such as geography, distance, and cost, influence real-world network designs. For example, while an ER graph might be resilient to targeted attacks, it might be very costly to build, or might not be doable given the geographic locations of the routers in the network.

Score from your peers: **1**

Comments: Please enter an explanation for your scoring, especially if you deducted any points for one of the rubric items for this question.

peer 1 → You should disagree with using either graphs for a real world network.

peer 2 → very exhaustive answer, good job

peer 3 → *[This area was left blank by the evaluator.]*

peer 4 → *[This area was left blank by the evaluator.]*

peer 5 → *[This area was left blank by the evaluator.]*