Name: Keith Pham

SID: 32507133

P2: PageRank

1. This code is handcrafted by me
2. The code flow follows the pseudo code from the book (Fig 4.11, pg.110) with some minor adjustments. There are several issues regarding the large file:
    a. Python default read file cannot decode UTF-8. As a result, encoding UTF-8 is added to read the *links.srt*
    b. The O(N^2) from pseudo code is not efficient. To optimize the run-time to O(N*M), I move the loop that add an additional probability if Q is empty outside; replace it with an *accumulator*; then add the *accumulator* to every page in P outside of the loop.
    c. The Map P stores string as key. This is not as efficient as storing an integer as key. However, to keep the code's simplicity, key string data structure is kept.
    d. A list of objects { 'url': page_name, 'rank/value: page_rank_value } is used to stored list of top 100 inlinks and pagerank
3. No external library is used for this project
4. The top 100 highest inlink count and pagerank are quite similar. Most pages that are in top inlink are also in top pagerank. However, there are some differences in ranking. Here is a scenario: page A is pointed by other 10 pages; the 10 pages do not have any other pages point to; On the other hand, page B is pointed by only 1 page C, but C is pointed by billions of pages.
    - The inlink count only considers the first layer of popularity. In this case, A is ranked higher than B in inlink count
    - The pagerank for B inherits some of the popularity of page C. As a result, B is ranked higher than A
5.
    - If we initialize the PageRank scores to random values instead, the algorithm will take longer or shorter (depending on the random values) to converge.
    - If we initialize the PageRank scores to all zeroes but still keep random surfer, the algorithm will eventually converge but much slower.
    - If we take away random surfer, which mean lamda = 0, then the algorithm may get stuck in a loop links or a dangling link. As a result, convergence is not guaranteed.