Web Crawling

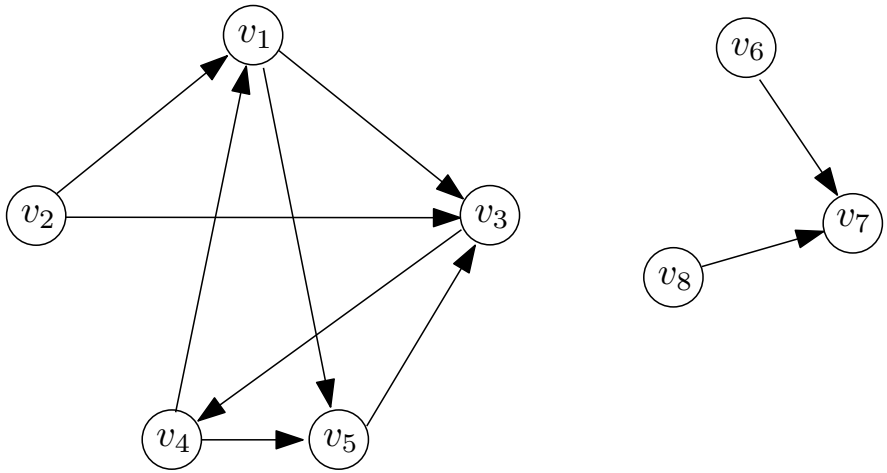
Yufei Tao

KAIST

March 26, 2013

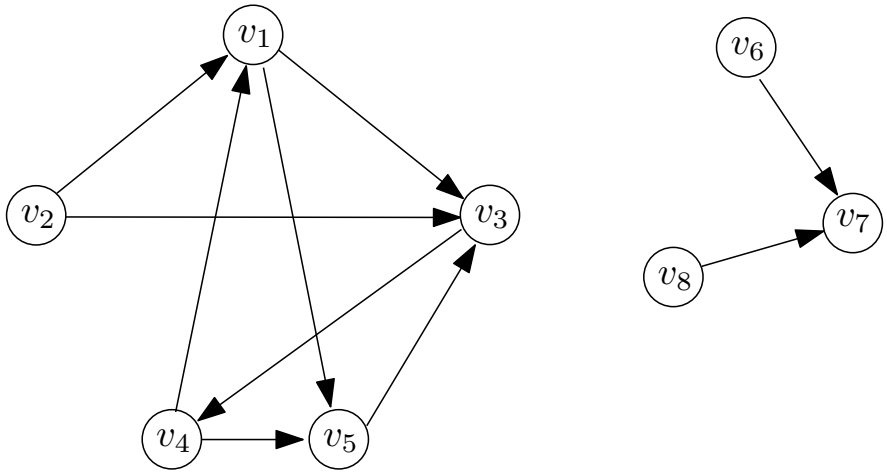
For a search engine to answer users' queries, it has to store all the webpages of the entire WWW in its local storage. For this purpose, obviously, the search engine will have to read every webpage once, a process known as web crawling. In this lecture, we will discuss how this can be done.

We will once again model WWW as a graph $G = (V, E)$. Specifically, every webpage corresponds to a vertex in V. If webpage u has a hyperlink to webpage v, E has an edge from vertex u to vertex v. The following is an example:



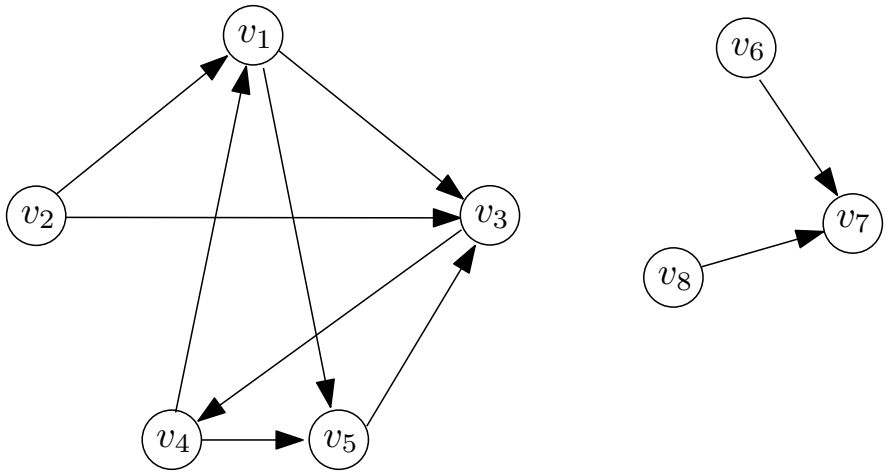
The goal of the web crawler is to visit all the webpages (a.k.a. vertices) in G exactly once. Let the beginning, the crawler knows the URLs of only some of the webpages (otherwise, crawling becomes a trivial task – why?). Let S, a subset of V, contain all the webpages whose URLs are known to the crawler.

In our example, let us assume $S = \{v_4, v_1, v_2, v_6\}$.



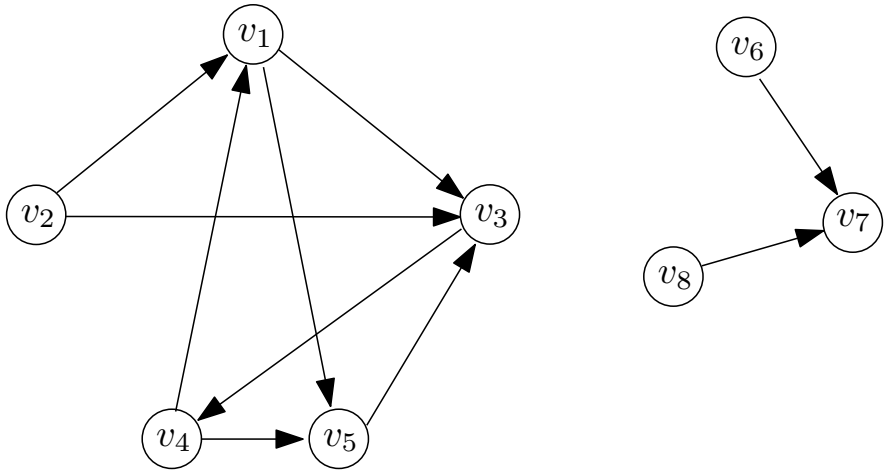
To start with, the crawler picks a vertex $v \in S$, and performs breadth first search (BFS) in G, to visit all the vertices that are reachable from v. Specifically, BFS works as follows:

algorithm BFS (v, A)
/* v is a vertex in S, and A is the set of vertices already visited so far */
1. let *queue* be a queue that contains only v
2. **while** *queue* is not empty
3. remove the first vertex u from *queue*
4. visit u, and add u to A
5. **for** every vertex u' that u has an edge to
6. **if** $u' \notin A$ and $u' \notin$ *queue* **then**
7. append u' to *queue*



Continuing our example, suppose that the crawler picks v_4 from S. The BFS starting from v_4 visits v_4, v_1, v_5, v_3 (in this order). Hence, when it finishes, $A = \{v_4, v_1, v_5, v_3\}$.

Next, the crawler discards from S all the vertices that are already in A. Then, it repeats the above procedure, until S becomes empty.



Continuing our example, the crawler removes v_4, v_1 from S. Suppose that the next vertex it picks from S is v_2. The BFS starting from v_2 visits only v_2 itself. Then, after removing v_2 from S, it picks the last vertex v_6 from S, and launches one more BFS from v_6, which visits v_6 and v_7.

At this moment, the algorithm finishes, and has visited $v_1, v_2, ..., v_7$, but not v_8.

In general, a web crawler can visit only a subset of the webpages in WWW.

It should be now clear that the effectiveness of crawling depends on how large S is. Fortunately, a sufficiently large S can be easily obtained by enumerating all the possible IP addresses. Such an S will allow the crawler to reach billions of pages in WWW, although it still does not guarantee reaching all of them (think: why?).