Problem 1:

A. How can the owner of a web site design a “spider trap”?  
   **Answer:** He can set up his web server so that, whenever a client requests a URL in a particular directory D, the server dynamically creates a new page containing links with newly created file names in D. Thus the crawler will request these pages; these will generate new names; and so on.

B. Is it possible for a crawler to distinguish between a spider trap and a legitimate web site with 100% certainty? Explain.  
   **Answer:** No. The crawler has no way to distinguish between a statically page and a dynamically generated page, either in terms of the URL or in terms of the content.

C. Describe one method that a crawler might use to avoid spider traps. What are the drawbacks of this method?  
   **Answer:** Crawlers impose a maximum on the total number of pages to be downloaded from any single site. Another rule would be to impose a maximum on the depth of any crawling path within a single site, but this is not secure; it can be defeated either by a crawler trap involving multiple sites, or by a crawler trap where each page downloaded contains an immense number of hyperlinks. The drawback is the crawler will only partial index genuinely useful sites with immense numbers of pages.

Problem 2: In the vector model of documents,

A. What are the dimensions of the vector space? What is the meaning of the coordinate of document D in dimension J?  
   **Answer:** Each word in the lexicon is a “dimension”. The coordinate of document D in dimension J is a measure of the “significance” of J within D; or, conversely, the “relevance” of document D to a query involving J, ignoring the absolute value of D.

B. In the TF-IDF model, how is the value of the coordinate of document D in coordinate J computed? (There are a number of different formulas that can be used; you may give any one of these.)  
   **Answer:** Suppose that word J appears M times in document D and that there are N documents in the collection and that J appears in K of these. Then the term frequency (TF) of J in D is M; the inverse document frequency (IDF) of J in D is log(N/K); and the TF-IDF measure of J in D is M*log(N/K). (There are also other formulas.)

C. What is the reason for including the IDF factor, instead of using just TF?  
   **Answer:** The rarer the word, the more it presumably indicates about the meanings of the query and of the document.

D. Actual Web search engines that use a vector model presumably use a modified version of the formula in B. Name three considerations that might enter into the value of the coordinates of document D in coordinate J. (Note: These must be considerations that are dealt with at indexing time not at query time, and that involve both the word and the document, not just the word or just the document.)  
   **Answer:** The font and size; whether the word appears in the URL, the title or other major headings (e.g. <H1>); whether the word appears in anchors of links pointing to the page or in the neighborhood of links pointing to the page.
Problem 3: Describe the operations involved at query time in computing a “relevance score” for a document as an answer to a multi-word simple query (no quoted strings, no Boolean operators, no special features) in Google. Try to include in your answer all the consideration that Google is known to use, or may be observed to use. For each of these, describe what data is used, when the data is computed, and what data structure it is stored in.

Answer:

• The relevance measure of the word within the document, as described in question 2. Computed for each document D and word W in D when D is indexed. Possibly updated when a new document that points to D is indexed. The value is saved in the inverted file, indexed under D and W.

• The pagerank of D. Pageranks for all documents are computed at the end of each cycle of crawling, and are recorded in the document index.

• For a multi-word query Q, whether the words in Q appear in D
  – In exactly the order given in Q; or
  – Close together.

  Computed either from the inverted file or from the cached copy of D; probably the former.

• If the match between the word in Q and the corresponding word in D relies on stemming, then some measure of the “stemming distance” between the two words. Presumably computed by the stemming algorithm.

Problem 4: Pages in the Web graph is divided into the categories SCC, IN, OUT, TENDRILS, and DCC.

A. How are these categories related to the WCC?
   Answer: SCC, IN, OUT, and TENDRILS are subsets of the WCC; in fact, their union is equal to the WCC. DCC is disjoint from WCC.

B. Suppose that a crawler starts with a seed set consisting of a single page in the SCC. Describe the categories of pages that the crawler can reach. (The form of your answer should be something like “All of the pages in DCC and IN and some of the pages in TENDRILS”.)
   Answer: All the pages in SCC and OUT; no other pages.

C. Repeat part (B) for a seed set consisting of a single page in IN.
   Answer: All the pages in SCC and OUT; some of the pages in IN; possibly some of the pages in TENDRILS.

D. Repeat part (B) for a seed set consisting of a single page in DCC.
   Answer: Only pages in DCC.

Problem 5: In computing PageRank, one solution to the problem of pages with no outlinks is to posit that every page is considered to have an implicit self-link (that is, a link to itself).
A. Using that assumption, set up the system of linear equations that defines the PageRank for
the graph shown below. (You need not compute the solution.) Assume that the parameter
E, as defined in class project 2, (the probability of following an outlink rather than jumping
randomly) is equal to 0.6.

\[
\begin{align*}
T &= 0.06667 + 0.15 \cdot U + 0.6 \cdot T \\
U &= 0.06667 + 0.15 \cdot U \\
V &= 0.06667 + 0.15 \cdot U + 0.2 \cdot V \\
W &= 0.06667 + 0.15 \cdot U + 0.2 \cdot V + 0.3 \cdot W \\
X &= 0.06667 + 0.2 \cdot V + 0.6 \cdot X \\
Y &= 0.06667 + 0.3 \cdot W + 0.6 \cdot Y
\end{align*}
\]

The value of the constant term in the above equations is not significant; replacing "0.06667"
in all the equations by a different value C would just change all the values in the solution by
C/0.06667. The only reason for choosing 0.06667 (= 1 - E/N where E = 0.6, the probability
of following an outlink and N = 6, the number of nodes) is so that the PageRanks add up to
1.0.

System 2: The PageRank of each node J is the sum over I of (the PageRank at I) times
(the probability, if you are at I, that you will transition to J). The latter term is the sum of
the probability that you will follow an outlink from I to J plus the probability that you will
randomly jump from I to J (which is always 0.4/6 = 0.06667)

\[
\begin{align*}
T &= 0.06667T + 0.216667U + 0.06667V + 0.06667W + 0.06667X + 0.06667Y \\
U &= 0.06667T + 0.216667U + 0.06667V + 0.06667W + 0.06667X + 0.06667Y \\
V &= 0.06667T + 0.216667U + 0.26667V + 0.06667W + 0.06667X + 0.06667Y \\
W &= 0.06667T + 0.216667U + 0.26667V + 0.36667W + 0.06667X + 0.06667Y \\
X &= 0.06667T + 0.06667U + 0.26667V + 0.66667W + 0.66667X + 0.06667Y \\
Y &= 0.06667T + 0.06667U + 0.06667V + 0.36667W + 0.06667X + 0.66667Y \\
1 &= T + U + V + W + X + Y.
\end{align*}
\]

Again, the value of the constant term in the last equation is unimportant, as long as it is not
equal to zero. The reason to include the last equation is to rule out the zero solution.

B. What does this approach (considering each node to have a self-link) not give good results?

**Answer:** It gives a great advantage to pages with no outlink, which endorse exclusively
themselves and thus (if the parameter E is large) build up a large, self-compounding page
rank. For example, if $E=0.85$ (the value proposed in the PageRank paper) then the PageRank of a page with no inlinks and no outlinks would be 6.6667 times greater than a page with no inlinks and many outlinks.

**Problem 6:** I have never seen it in print, but I think that it is quite safe to assume that Google keeps a log of the queries submitted; that the frequency of specific queries follows an inverse power-law (Zipf) distribution; and that Google caches the answers to the most frequent queries in some fast memory, and these are answered without need to refer to the main inverted index.

A. There is a feature of the Zipf distribution that implies that it will be reasonably easy to achieve a fairly large cache hit rate. Explain.

**Answer:** In a Zipf distribution, a fairly small fraction of the most popular questions account for a large fraction of the user queries.

B. There is a different feature of the Zipf distribution that implies that it will be very difficult to achieve cache hit rates that are close to 100%. Explain.

**Answer:** A Zipf distribution has a “long tail” meaning that very rare questions (questions that appear only once in the log) account a non-negligible fraction of the user queries.

C. How could knowing the shape of this distribution be useful in tuning the structure of the cache? I don’t mean ”knowing which queries are most frequent” which you obviously need; I mean knowing a fact like, “The $N$th most frequent query has a frequency proportional to $1/N^{1.7}$.”

**Answer:** Presumably there is some trade-off between making this cache large and other desirable features such as running time for queries not in the cache. Knowing the shape of the distribution could help one choose an optimal size for the cache; that is, finding the point at which making the cache larger would cost you more for cache misses than it saves you for cache hits.

**Part II** (40 points)

**Problem 7:** (Open-ended. Your answer should be no more than about 1000 words.) Suppose that you wanted to automate the following task using web resources: Given a subject $Q$, and a starting date and an ending date, construct a chronology of the major events involving $Q$ between the starting and ending date.

A. Propose a method for carrying out this task. Your method may use any existing online resources you choose. If you want, there can be a human in the loop, but, of course, some significant part of the task must be done by the computer.

B. Propose a learning method that would enable part (A) to become more effective over time.

C. How would you evaluate your program? What measure of quality would you use? How could you operationalize this measure? (You may use either human or automated evaluators.)
ONE POSSIBLE ANSWER:

Execution method:

Task execution involves the following steps:

**Step 1:** Document collection. Issue queries such as Q; Q with "history"; Q with "chronology" and so on to a general search engine. Collate the first K answers from each, removing duplicates, and giving some degree of priority to those that came from the query with "history" etc. over those that just came from the query Q. Download all the selected documents.

**Step 2:** Event identification. Find all sentences that contain both what appears to be a date between the starting and ending value and a word in Q. Each such sentence is a candidate “event”. Of course, this method is highly susceptible, both to errors of omission (recall) and errors of commission (precision); but hopefully, the errors of omission will be handled by the redundancy of the web (some other document will describe this event in a way we can recognize) while errors of commission will be weeded out in the ranking procedure below.

**Step 3:** Event collation. The problem is to identify different descriptions of the same event. This may well be the hardest step of the whole thing. The likelihood that two sentences describe the same event is an increasing function of (A) the degree to which the associated dates are identical, compatible (i.e. "1864" and "March 3, 1864"), or close ("March 1864" and "April 1864"); (B) the degree to which the sentence describing the events contain the same words or words known to be related (e.g. marked as synonymous in an online thesaurus.) Partition the event sentences into equivalence classes based on these likelihoods. Again, there will be both kinds of errors, though I think it is safe to guess that this will fail to identify events more often than it incorrectly identifies events, leading to substantial oversplitting. The conflation of two events will be largely solved by step 5, whereas the splitting of the same event will to some extent be reduced by step 4.

**Step 4:** Ranking. Events are ranked in order of importance by such considerations as:

- The number and quality of the documents that mention the event.
- The prominence of the event in the document. For instance, is the name of the event a section heading?
- Inclusion of the event in an HTML list labelled "History", "Chronology" etc.
- Preset categories of important events. E.g. if Q is a person then birth, death, marriage, election to public office and so on are important events. If Q is a country then wars, revolutions, and constitutions are important events.

**Step 5:** Best sentence. One sentence out of the collection associated with an event is chosen as the best representative. This uses such criteria as

- Specificity of date (the more specific, the better).
- Prominence of the query term (best if the query words are the first words in the sentence, or close to the front.)

Learning Method

The execution method outlined above actually does not seem to present opportunities for many different types of learning.
One easy type of learning that can always be done is parameter tuning. There are lots of fairly arbitrary real-valued parameters in the above algorithm, (weights on different features being combined etc.) If you can get a human evaluator to go through the intermediate steps of calculation over a collection of instances and label where the various steps have done the right thing and the wrong thing, then it is possible to tune the parameter values to get optimal results. The problem is that parameter tuning is not usually very effective.

Another possibility would be to look for more keywords, if one has collected sets of good web pages and good events. You have different keywords for different functions: For instance the presence of "biography" or "timeline" on the page (particularly prominently) might indicate that this is a good page, whereas other keywords that indicate either a specific type of event or general importance (e.g. "important") might indicate that this is a good sentence.

### Evaluation

First, let us categorize the kinds of possible errors:

- A. An important event is missing.
- B. The event is misdated.
- C. The event in the sentence never occurred. (It was hypothetical, alleged, imaginary, feared, or the web page is just mistaken.)
- D. The event did not involve the subject of the query.
- E. The event was not important as regards the subject of the query.
- F. The sentence does not describe an event. (This can easily happen if the number being interpreted as a year is in fact some other kind of number.)
- G. The same event appears twice on the list.

So a simple dichotomy of recall vs. precision does not really fit this application very well. In particular errors of form (G) are not really either errors of precision or recall. If you do want to force this into the recall vs. precision framework, the best thing would probably be to consider type (A) to be errors of recall; types (B) through (F) to be errors of precision; and to ignore type G.

A human evaluator will be able to detect most instance of errors (F) and (G) and some instances of B, C, D, and E just looking at the results page. If the human evaluator looks at the web pages that are the source of the information, or even the immediate neighborhood of the sentence quoted on the results page, he/she will be able to find almost instances of all errors B through G.

Errors of type (A) — recall errors — are, as always, more difficult. The best solution would probably be to ask an expert to generate a list or find such a list in an authoritative source. Manually check to see what fraction of the expert’s list have been found on the computer’s list.

Automated evaluation is much harder and much less accurate. The best I can think of would be this: Use a high-precision, low-recall filter to gather from the web chronologies of important events whose subject is clearly marked. Run our program on that subject, deliberately excluding the these high quality chronologies. As best as one can, try to match the events in the chronology against events in our results list. Of course, this kind of evaluation has all kinds of problems:

- The class of subjects with high-quality chronologies on the web is not a representative sample.
• By excluding the high-quality chronology, the evaluation methodology is making things unrealistically difficult for the program.

• The automated matching cannot be made very precise.

In practice, the third problem is likely to be so large as to make the results largely meaningless as absolute values. It is possible, though, that this methodology could be used for comparative studies (determining whether one date-extractor was better than another.)