CAIM, examen final

20 de gener de 2016. Temps: 2 hores 50 minuts

Exercise 1

(2 points) Suppose that we have a collection of documents where the term `world` appears in $2 \cdot 10^5$ documents, the term `Trump` appears in $1.5 \cdot 10^5$ documents and the term `disaster` appears in $2 \cdot 10^4$ documents. What would be the best processing ordering for the query

```
Trump AND world AND disaster
```

based on a worst case analysis? Assume that the `AND` operations involve a merge algorithm of posting lists.

Answer of exercise 1

There are only $\binom{3}{2} = 3$ orderings:

1. `(Trump AND world) AND disaster`.
2. `Trump AND (world AND disaster)`
3. `(Trump AND disaster) AND world`

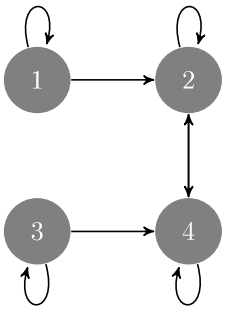
We define c_{inner} and c_{outer} as the worst case time cost of the inner and the outer AND, respectively. The total worst case time cost is $c = c_{inner} + c_{outer}$. The cost of every ordering is

1. $c = 520000$ with $c_{inner} = 350000$ and $c_{outer} = 170000$.
2. $c = 390000$ with $c_{inner} = 220000$ and $c_{outer} = 170000$.
3. $c = 390000$ with $c_{inner} = 170000$ and $c_{outer} = 220000$.

We conclude that the 1st ordering should be discarded. The other two orderings give the same worst case cost.

Exercise 2

(2 points) Consider a small web defined by the following graph



1. Give the PageRank equations and the PageRank weights of each node as a function of the damping factor λ.
2. Give the PageRank weights for $\lambda = 3/4$.

Answer of exercise 2

1. To obtain the PageRank weights, we could calculate the Google matrix and use it to produce the equations for each weight. We choose a shorter path based on a definition of p_i, the PageRank weight of the i-th vertex, that incorporates the damping factor λ as

$$p_i = \frac{1-\lambda}{n} + \lambda \sum_{j \to i} \frac{p_j}{k_j^{out}}, \tag{1}$$

where k_j^{out} is the out degree of the j-th vertex. In our network, we have $n = 4$ and $k_j^{out} = 2$ for all vertices. Applying Eq. 1, we obtain four PageRank equations, namely

$$p_1 = \frac{1-\lambda}{4} + \lambda \frac{p_1}{2} \tag{2}$$

$$p_2 = \frac{1-\lambda}{4} + \frac{\lambda}{2}(p_1 + p_2 + p_4) \tag{3}$$

$$p_3 = \frac{1-\lambda}{4} + \lambda \frac{p_3}{4} \tag{4}$$

$$p_4 = \frac{1-\lambda}{4} + \frac{\lambda}{2}(p_2 + p_3 + p_4) \tag{5}$$

Applying the customary normalization condition yields the 5-th PageRank equation, namely

$$p_1 + p_2 + p_3 + p_4 = 1. \tag{6}$$

Easily,

$$p_1 = p_3 = \frac{1-\lambda}{2(2-\lambda)}.$$

Applying Eq. 6 to to Eqs. 3 and 5, we obtain

$$p_2 = \frac{1-\lambda}{4} + \frac{\lambda}{2}(1 - p_3)$$
$$p_4 = \frac{1-\lambda}{4} + \frac{\lambda}{2}(1 - p_1).$$

Now it has become obvious that $p_2 = p_4$ because $p_1 = p_3$. Applying Eq. 1 to Eq. 7, one obtains

$$p_2 = p_4 = \frac{1}{4-2\lambda}$$

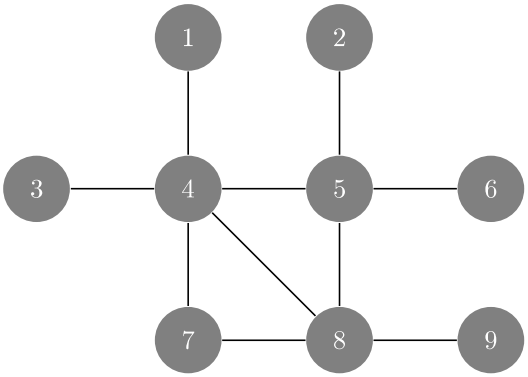
after some algebra.

2.

$$p_1 = p_3 = 1/10 = 0.1$$
$$p_2 = p_4 = 2/5 = 0.4.$$

Exercise 3

(2 points) We wish to analyze the clustering of the following network



1. Calculate the global clustering coefficient (C) and the transitivity coefficient (T). We define the global clustering as the mean local clustering and the transitivity coefficient as the proportion of connected triples of vertices that are closed.

2. Interpret the difference between C and T for the networks above.

3. Compare the clustering of the network against the expected clustering of the corresponding Erdős-Rényi graph.

Answer of exercise 3

1. Suppose that C_i is the local clustering of the i-th vertex and k_i is its degree. Recall that the local clustering coefficient is defined as

$$C_i = \frac{m_i}{\binom{k_i}{2}} = \frac{2m_i}{k_i(k_i-1)},$$

where m_i is the number of different pairs of neighbours of i that are connected. We adopt the convention that $C_i = 0$ when $k_i < 2$. Thus, $C_1 = C_2 = C_3 = C_6 = C_9 = 0$. We also have

$$C_4 = \frac{2}{\binom{5}{2}} = \frac{1}{5}; \quad C_5 = \frac{1}{\binom{4}{2}} = \frac{1}{6}$$
$$C_7 = \frac{1}{\binom{2}{2}} = 1; \quad C_8 = \frac{2}{\binom{4}{2}} = \frac{1}{3}$$

and thus

$$C = \frac{1}{9}\left(\frac{1}{5} + \frac{1}{6} + 1 + \frac{1}{3}\right) \tag{7}$$
$$= 17/90 \approx 0.19. \tag{8}$$

T is the number of closed triples of vertices over the total number of triples. The total number of triples is defined as the union of sets of triples that have a certain node at the center of the triple. Notice that

- Vertex 4 yields $\binom{5}{2}$ of which only 2 are closed.
- Vertex 5 yields $\binom{4}{2}$ of which only 1 is closed.
- Vertex 7 yields $\binom{2}{2}$ and 1 that is closed.
- Vertex 8 yields $\binom{4}{2}$ of which only 2 are closed.

Therefore,

$$T = \frac{2+1+1+2}{\binom{5}{2} + 2\binom{4}{2} + \binom{2}{2}} \tag{9}$$
$$= \frac{6}{23} \approx 0.26. \tag{10}$$

2. Every vertex contributes equally to clustering in C whereas the contribution of every vertex depends on its degree in T. C is smaller than T because the value of C is dominated by many vertices for which $C_i = 0$. In turn, this is due by the abundance of vertices of degree 1.

3. The clustering that is expected for an Erdős-Rényi graph coincides with the network density of links, namely

$$\delta = \frac{m}{\binom{n}{2}},$$

where m is the number of edges and n that of vertices. $m = 10$ and $n = 9$ give

$$C_{ER} = 5/18 \approx 0.28.$$

We find that C is about 1.5 times larger than C_{ER} while T is about the same as C_{ER}.

Exercise 4

(2 points) We wish to calculate the local clustering of every vertex of a large real network using the MapReduce framework. Suppose that the network is undirected and that vertices are labelled with integers from 1 to n. As input, we are given a graph implemented as an adjacency list A, i.e. for every vertex i, we are given l_i, a *set* of integers between 1 and n containing the vertices that are adjacent to i. We can consider that A is a dictionary where the keys are integers (vertices) and the values are adjacency sets. As output, we have to produce a list of pairs (i, C_i), where C_i is the local clustering coefficient of vertex i. Recall that C_i is the proportion of different pairs of neighbours of i that are connected.

Provide pseudo code for map and reduce functions and, if appropriate, combine and partition functions. It is possible to use more than one map-reduce phase if required.

Answer of exercise 4

The local clustering coefficient is defined as

$$C_i = \frac{m_i}{\binom{k_i}{2}} = \frac{2m_i}{k_i(k_i-1)},$$

where m_i is the number of different pairs of neighbours of i that are connected and k_i is the degree of i. When $k_i < 2$, we adopt the convention that $C_i = 0$. A possible solution follows.

Every instance of `map` receives a vertex i, its adjacency set s (i.e. l_i), and B, a subset of the dictionary A where the entries correspond only to the neighbours of i.

```
map(i, s, B)
  k = size(s) // k is the degree of i
  if k < 2 output((i, k), 0)
  else
     for each distinct pair of vertices (u, v) in s
        x is a boolean (0 or 1) that indicates if (u, v) are linked according to B
        if x then output((i, k), x)

combine((i, k), L)
  output((i, k), sum(L))

reduce((i, k), L)
  if k < 2 emit(i, 0)
  else
     m = sum(L)
     pairs = k(k-1)/2
     emit(i, m/pairs)
```

Exercise 5

(1 point)

1. Explain the cold start problem in recommender systems. Provide a general definition and explain how it concerns recommendation methods. Do not use more than six sentences.

2. Tell a typical application of locality sensitive hashing in information retrieval. Use only one sentence.

Answer of exercise 5

1. The cold start problem in recommender systems is the problem of lacking the information that is needed to be able to start making recommendations. The depth of the problem depends on the method. Within collaborative filtering, the problem arises when a new user or a new item arrives. It is not possible to offer recommendations to a user who has not provided ratings or equivalent information. Similarly, it is not possible to recommend a new product when ratings or equivalent information are not available. Within content-based methods, the cold start problem disappears or reduces for items but remains for users.

2. A typical application of locality sensitive hashing is the detection of near-duplicated documents (e.g., web pages).

Exercise 6

(1 point) Suppose that we have a family of hash functions such that for any two objects x and y, the probability of collision satisfies

$$P[h(x) = h(y)] = s(x, y), \tag{11}$$

where h is a member of that family and $s(x, y)$ is the similarity between x and y. Recall that a family of locality sensitive functions is (s, cs, p_1, p_2)-sensitive if for any two objects x and y we have

- If $s(x, y) \geq s$ then $P[g(x) = g(y)] \geq p_1$.
- If $s(x, y) \leq cs$ then $P[g(x) = g(y)] \leq p_2$.

with $c < 1$ and $p_1 > p_2$ and g being a locality sensitive function of the family. Explain why by stacking k hash functions and repeating the process m times it is possible to obtain a family of locality sensitive hashing functions that is (s, cs, p_1, p_2)-sensitive with

$$p_1 = 1 - (1 - s^k)^m$$

and

$$p_2 = 1 - (1 - (cs)^k)^m.$$

Answer of exercise 6

We have m tables. The output of k hashing functions that are chosen independently is concatenated to produce a position in one of the tables for an object x. Suppose that the tables are large enough so that the position that is produced by simple concatenation is valid. Thus the probability of a collision with another object y stored in the same table is $s(x, y)^k$ thanks to Eq. 11 and independence. The probability of a collision between x and y in at least one of the m tables is

$$\pi = 1 - (1 - s(x, y)^k)^m. \tag{12}$$

To see it, notice that $1 - s(x, y)^k$ is the probability of not colliding in one of the tables and thus $(1 - s(x, y)^k)^m$ is the probability of not colliding in any of the tables. By Eq. 12, if $s(x, y) \geq s$ then $P[g(x) = g(y)] \geq 1 - (1 - s^k)^m$ while if $s(x, y) \leq cs$ then $P[g(x) = g(y)] \leq 1 - (1 - (cs)^k)^m$. Notice that $1 - s^k$ increases as s increases ($0 < s < 1$) and thus $1 - (1 - s^k)^m$ also increases as s increases.