MDA HW2 Report

102062315 洪若恒

1.

程式在執行的時候，必須要先手動設好 pageNum 和 loopNum，才能正確地執行。此外，由於大測資是由 Page 0 開始，所以我也是由 0 作為第一個 Page。

```
public static int pageNum = 10878;
public static double beta = 0.8;
int loopNum = 20;
```

2.

我使用了三個 MapReduce。

首先，我的第一個 MapReduce 主要的工作是先建立一個矩陣，而這個矩陣會記錄每個 Page 連出去的數量和連到哪個 Page，並把連出去的機率 1 平均分配給那些被連到的 Page。

先由 Mapper 來依據 input 的 file 做出對應的(key, value)，再由 Reducer 去記住每個相同的 key 會連到哪些 value，並計算出連出去的數量，之後就能平均把機率分配出去。

這裡要注意，由於沒辦法開一個超過 10000*10000 的二維矩陣，所以我只有在有連出去的 edge 的情況下才把值寫出去，而沒有的情況就忽略不寫。

接著，我把第一次做出來的結果或是上一次跑的結果再拿來做另一個 MapReduce，重複 loopNum 次之後，在 renormalize。

做法和第一次作業相同，把矩陣的每一列和 r 相乘。如果讀進來的 value 是矩陣，則把(key, value)分別設為他所在的那一列和他在矩陣所在的位置的行列值；如果是 PageRank 結果的話，就要放到每一組列。

再來是 renormalize。我在 reduce 的時候，會利用兩個 hashmap 把 A 和 r 分別寫入。此時，由於 r 還會變動所以先不寫出，而因為我們需要原先 A 矩陣的中的值，所以會將它寫出來，接著進行矩陣乘法。之後，乘上 beta 後加上(1-beta)/pageNum，再以 r 所在的列為 index 存起來。

```
HashMap<Integer, Double> hashA = new HashMap<Integer, Double>();
HashMap<Integer, Double> hashR = new HashMap<Integer, Double>();
```

最後，則是把經過 loop 後並排序，再把答案以小數點後六位給秀出來。

```
record[page_index] = 1;
DecimalFormat output_form = new DecimalFormat("#.######");
String output = output_form.format(final_out[page_index]);
context.write(new Text(Integer.toString(page_index)), new Text(output));
```

```
1056    0.000632
1054    0.000629
1536    0.000524
171 0.000512
453 0.000496
407 0.000485
263 0.00048
4664    0.00047
261 0.000463
410 0.000461
```