L A S T H O M E W O R K

CS 667 : Homework 5(Due: Apr 25, 2013)

Problems 1-6 are for 200pts. You may replace some of them with Problem 7 or 8 for a total of 200.

Problem 1. (40 points)

(a) We plan to sort n keys using p processors ala-PRAM like using merge-sort as follows. We split the n input keys into p subarrays each one of n/p keys (don't worry about ceilings or floors). Each processor deals with one of those subarrays. At the end we end up with p sorted sequences of n/p keys. One processor takes over to complete the sorting.

(i) If $p = \sqrt{n}$, fill-in the details by providing the parallel running-time of the approach T and the speedup s achieved over regular merge-sort. Explain and justify your answers.

(ii) If $p = \lg n$, repeat the questions of part (i).

(b) Instead of using merge-sort we use bubble-sort of the sorting of the p sequences. Repeat questions (i) and (ii) above.

Problem 2. (40 points)

Let $S = \langle x_1, x_2, \ldots, x_n \rangle$ be a sequence of n distinct keys. The rank of x_1 in the sequence S or $r(x_1, S)$ is the number of keys less than x_1 in S. The problem of sorting is equivalently the problem of determining the rank of each one of the n input keys.

Determine the rank of all keys in S in $O(\lg n)$ time with a CRCW PRAM. How many processors did you use?

Sort the n keys in the same time with a CRCW PRAM. How many processors did you use?

Can you repeat the two questions above for an EREW PRAM? How would the answers change? Explain.

Problem 3. (40 points)

(a) Give an EREW PRAM algorithm that merges two sorted arrays of size $n/2$ with $P = n/2$ in $O(\lg n)$ time. You may assume that n is a power of two, and you may of course reuse prior or more recent results.

(b) Can you find the MAX of n keys with $n^{7/6}$ processors in $\Theta(1)$ time? Explain.

Problem 4. (40 points)

We execute one query in Google and Bing. There are two pages of results for each. The query is `CS 667 Algorithms`. There a total of 20 results in two pages per search engine. Relevant documents are those that can positively identified from the available information (title, URL, context) as OUR COURSE. Everything else is NOT relevant.

(a) For each search engine, find and give the number of hits reported by each engine. Give the number of relevant document (read previous paragraphs) out of the 20 listed. Give the precision relative to the 20 documents reported for each engine. (These are the first 3 items in the table below.)

(b) Give 6-point effectiveness along the lines of page 33 of Handout 6 by generating a table similar to that of page 33. (This is item 4 in the table below. Items 5-8 can also be extracted.)

(c) Fill the table below. One point for a winner and 0 for the loser, 1 each for a tie. Who is the winner? Tie ?

	Values: Google	Values: Bing	Points: Google	Points: Bing
1. # Number of Hits reported (question (a))				
2. # Number of relevant docs among the 20				
3. Precision among the 20				
4. 6-point effectiveness				
5. 20% recall interpolated precision				
6. 60% recall interpolated precision				
7. 80% recall interpolated precision				
8. 3-point effectiveness(20,60,80)				
Number of point wins (sum)	-------	------		

Problem 5. (20 points)

Kleinberg. Find the hub/authority rank of the graph of Figure 1. Initial values will be $1/N$ (not 1). Iterate as many times as needed for the error to be less than 10^{-4}. (Do not forget scaling.)

Problem 6. (20 points)

PageRank. Find the page rank of the graph of Figure 1. Initial values are $1/N$. Iterate as many times as needed for the error to be less than 10^{-4}.

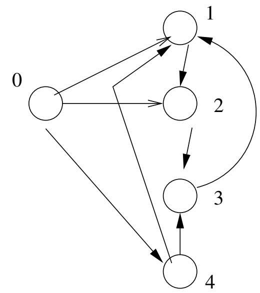


Figure 1: Problem 5-6 figure

Problem 7. (60 POINTS)

Use multithreading/multiprocessing, if you know how to do it, to implement the algorithm outlined in Problem 1 part (b). If you can't figure out how to implement bubble-sort, grab the code from my CS 435 web-page (section B4) and modify it as needed. I won't be able to help on multithreading. The data type used for testing would be randomly distributed doubles as in Homework 1. Time the running time of the sorting function implementation by capturing a running-time of ordinary bubble-sort, and then the modified one ON THE SAME INPUT sequence. A minimal interface needed is as follows.

```
% ./psort  threads nkeys
% java psort thread nkeys
```

Problem 8. (60 POINTS)

Implement the HITS and PageRank algorithms. The inputs will be graphs represented through an adjacency list. The command-line interface would be as follows.

```
% ./rank    ranktype   InitialValue Iterations InputFile
% java rank ranktype   InitialValue Iterations InputFile
```

The command-line parameter `ranktype` takes one of two values: `0` if Kleinberg's HITS is used (with the scaling as otherwise shown on page XX of Subject YY) and `1` if the Brin and Page's PageRank algorithm is used (as shown on page ZZ of Subject YY). The second parameter `InitialValue` indicates how the initial values for the ranks will be computed. If it is `0` all ranks are initialized to 0, if it is `1` they are initialized to 1. If it is `2` they are initialized to $1/N$, where N is the number of web-pages (size of the graph.) If the value is a numeric integer value other than 0,1,2 then the ranks are initialized as `InitialValue` divided by 100. Thus an `InitialValue` equal to 50, initializes all ranks to $50/100 = 0.5$. Parameter `Iterations` runs the algorithms for that number of iterations. Parameter `InputFile` describes the input graph and it has the following form. The first line contains two numbers: the number of vertices (in the example below, this is equal to five) and the number of edges that follow on separate lines (i.e. six). In each line an edge (i, j) is presented by `i j`. The graph used in class in a lecture will be represented as follows. (Note that the graphs in class have vertices in the range $1..n$, whereas in this implementation, it is $0..n-1$.

```
4 4
0 2
0 3
1 0
2 1
```

```
Kleinberg  might report, at the  14-th iteration, Authority/Hub pair values of


Base   :  0 :A/H[ 0]=0.25000/0.25000 A/H[ 1]=0.25000/0.25000 A/H[ 2]=0.25000/0.25000 A/H[ 3]=0.25000/0.25000
Iterat :  1 :A/H[ 0]=0.50000/0.81650 A/H[ 1]=0.50000/0.40825 A/H[ 2]=0.50000/0.40825 A/H[ 3]=0.50000/0.00000
Iterat :  2 :A/H[ 0]=0.31623/0.94281 A/H[ 1]=0.31623/0.23570 A/H[ 2]=0.63246/0.23570 A/H[ 3]=0.63246/0.00000
Iterat :  3 :A/H[ 0]=0.17150/0.98473 A/H[ 1]=0.17150/0.12309 A/H[ 2]=0.68599/0.12309 A/H[ 3]=0.68599/0.00000
Iterat :  4 :A/H[ 0]=0.08771/0.99612 A/H[ 1]=0.08771/0.06226 A/H[ 2]=0.70165/0.06226 A/H[ 3]=0.70165/0.00000
Iterat :  5 :A/H[ 0]=0.04411/0.99902 A/H[ 1]=0.04411/0.03122 A/H[ 2]=0.70573/0.03122 A/H[ 3]=0.70573/0.00000
Iterat :  6 :A/H[ 0]=0.02209/0.99976 A/H[ 1]=0.02209/0.01562 A/H[ 2]=0.70676/0.01562 A/H[ 3]=0.70676/0.00000
Iterat :  7 :A/H[ 0]=0.01105/0.99994 A/H[ 1]=0.01105/0.00781 A/H[ 2]=0.70702/0.00781 A/H[ 3]=0.70702/0.00000
Iterat :  8 :A/H[ 0]=0.00552/0.99998 A/H[ 1]=0.00552/0.00391 A/H[ 2]=0.70709/0.00391 A/H[ 3]=0.70709/0.00000
Iterat :  9 :A/H[ 0]=0.00276/1.00000 A/H[ 1]=0.00276/0.00195 A/H[ 2]=0.70710/0.00195 A/H[ 3]=0.70710/0.00000
Iterat : 10 :A/H[ 0]=0.00138/1.00000 A/H[ 1]=0.00138/0.00098 A/H[ 2]=0.70711/0.00098 A/H[ 3]=0.70711/0.00000
Iterat : 11 :A/H[ 0]=0.00069/1.00000 A/H[ 1]=0.00069/0.00049 A/H[ 2]=0.70711/0.00049 A/H[ 3]=0.70711/0.00000
Iterat : 12 :A/H[ 0]=0.00035/1.00000 A/H[ 1]=0.00035/0.00024 A/H[ 2]=0.70711/0.00024 A/H[ 3]=0.70711/0.00000
Iterat : 13 :A/H[ 0]=0.00017/1.00000 A/H[ 1]=0.00017/0.00012 A/H[ 2]=0.70711/0.00012 A/H[ 3]=0.70711/0.00000
Iterat : 14 :A/H[ 0]=0.00009/1.00000 A/H[ 1]=0.00009/0.00006 A/H[ 2]=0.70711/0.00006 A/H[ 3]=0.70711/0.00000

and PageRank
Base   :  0 :P[ 0]=0.25000 P[ 1]=0.25000 P[ 2]=0.25000 P[ 3]=0.25000
Iter   :  1 :P[ 0]=0.25000 P[ 1]=0.25000 P[ 2]=0.14375 P[ 3]=0.14375
Iter   :  2 :P[ 0]=0.25000 P[ 1]=0.15969 P[ 2]=0.14375 P[ 3]=0.14375
Iter   :  3 :P[ 0]=0.17323 P[ 1]=0.15969 P[ 2]=0.14375 P[ 3]=0.14375
Iter   :  4 :P[ 0]=0.17323 P[ 1]=0.15969 P[ 2]=0.11112 P[ 3]=0.11112
Iter   :  5 :P[ 0]=0.17323 P[ 1]=0.13196 P[ 2]=0.11112 P[ 3]=0.11112
Iter   :  6 :P[ 0]=0.14966 P[ 1]=0.13196 P[ 2]=0.11112 P[ 3]=0.11112
Iter   :  7 :P[ 0]=0.14966 P[ 1]=0.13196 P[ 2]=0.10111 P[ 3]=0.10111
Iter   :  8 :P[ 0]=0.14966 P[ 1]=0.12344 P[ 2]=0.10111 P[ 3]=0.10111
Iter   :  9 :P[ 0]=0.14242 P[ 1]=0.12344 P[ 2]=0.10111 P[ 3]=0.10111
Iter   : 10 :P[ 0]=0.14242 P[ 1]=0.12344 P[ 2]=0.09803 P[ 3]=0.09803
Iter   : 11 :P[ 0]=0.14242 P[ 1]=0.12083 P[ 2]=0.09803 P[ 3]=0.09803
Iter   : 12 :P[ 0]=0.14020 P[ 1]=0.12083 P[ 2]=0.09803 P[ 3]=0.09803
Iter   : 13 :P[ 0]=0.14020 P[ 1]=0.12083 P[ 2]=0.09709 P[ 3]=0.09709
Iter   : 14 :P[ 0]=0.14020 P[ 1]=0.12002 P[ 2]=0.09709 P[ 3]=0.09709
Iter   : 15 :P[ 0]=0.13952 P[ 1]=0.12002 P[ 2]=0.09709 P[ 3]=0.09709
Iter   : 16 :P[ 0]=0.13952 P[ 1]=0.12002 P[ 2]=0.09680 P[ 3]=0.09680
Iter   : 17 :P[ 0]=0.13952 P[ 1]=0.11978 P[ 2]=0.09680 P[ 3]=0.09680
Iter   : 18 :P[ 0]=0.13931 P[ 1]=0.11978 P[ 2]=0.09680 P[ 3]=0.09680
Iter   : 19 :P[ 0]=0.13931 P[ 1]=0.11978 P[ 2]=0.09671 P[ 3]=0.09671
```

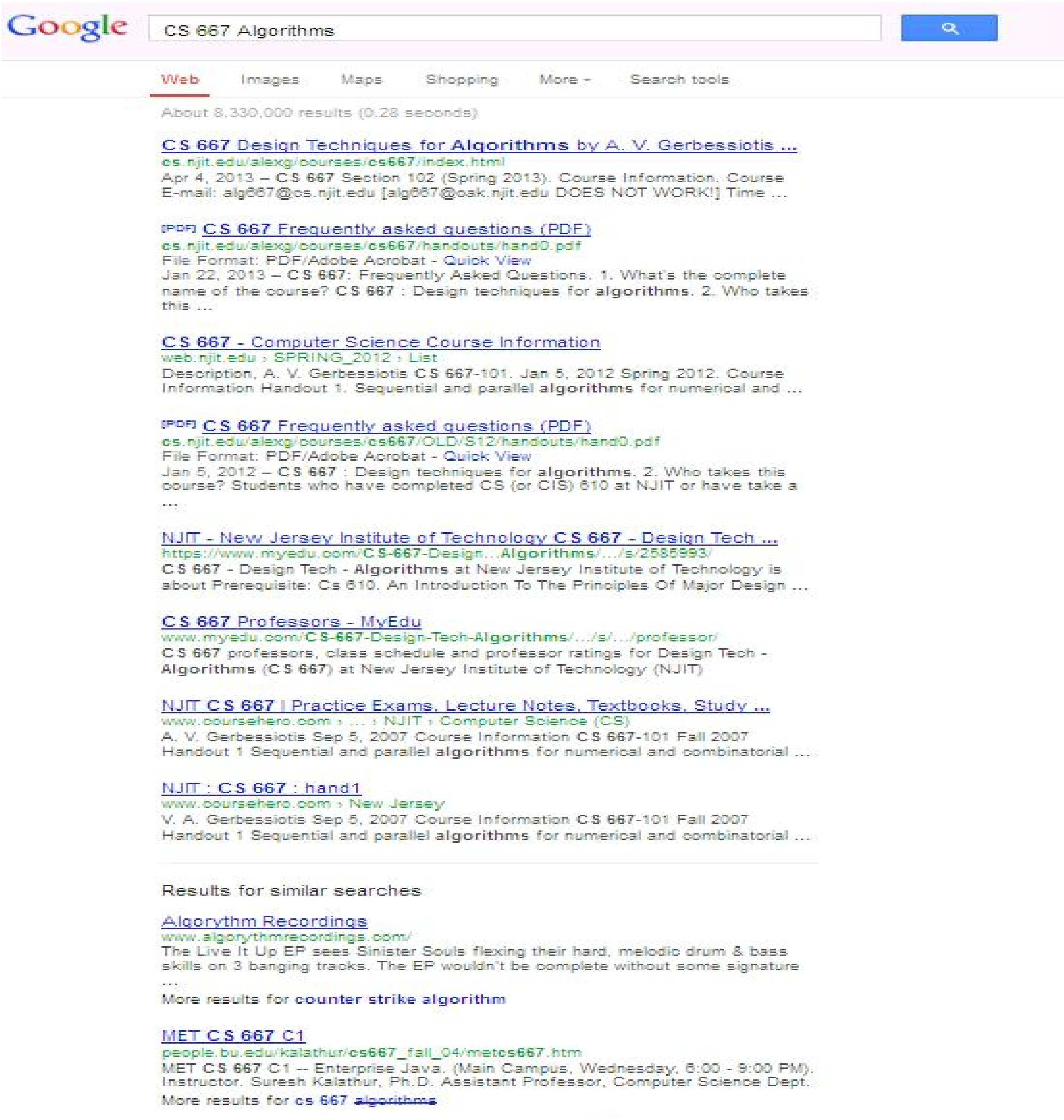


Figure 2: Google for `CS 667 Algorithms`

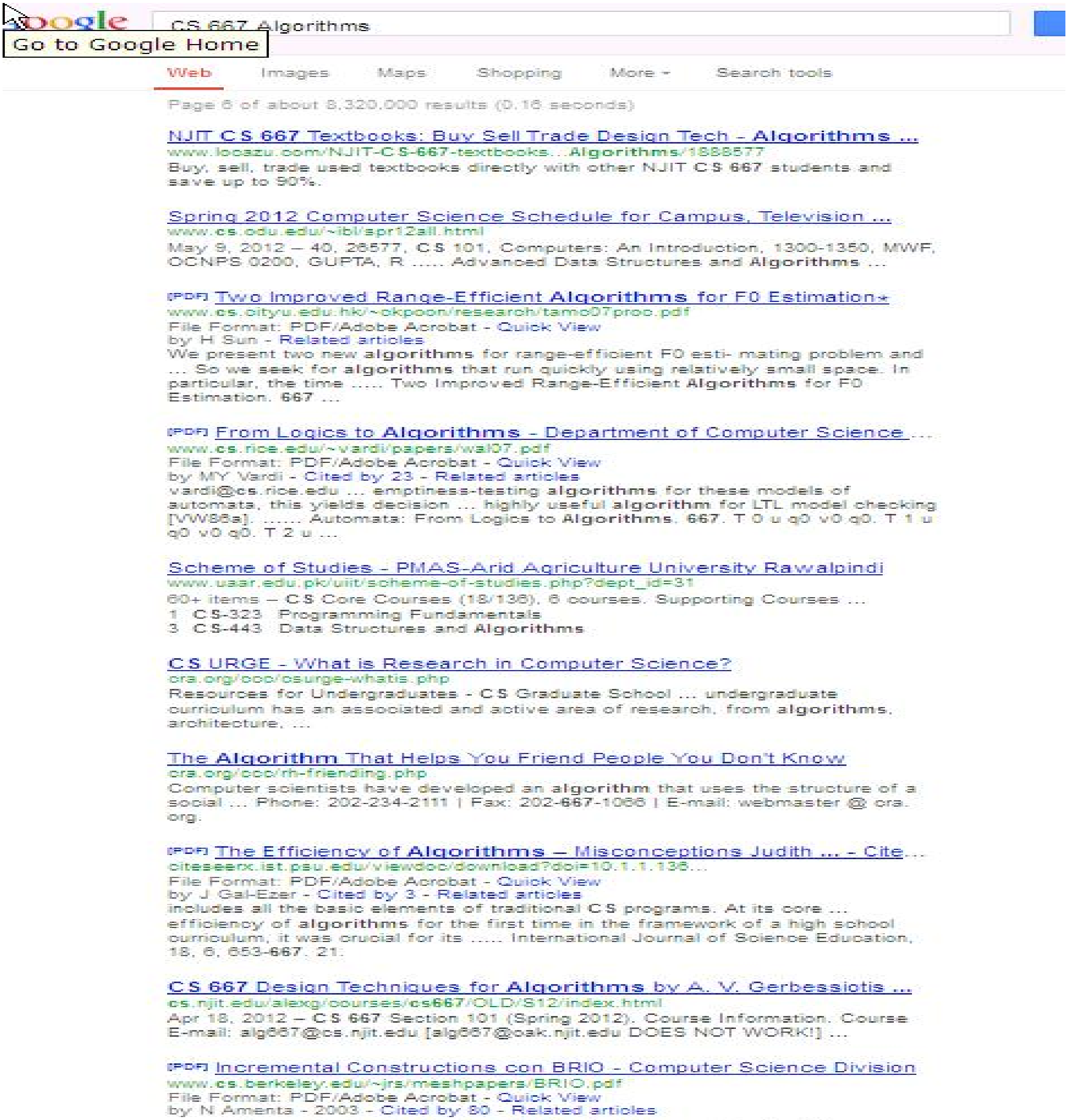


Figure 3: Google for `CS 667 Algorithms`

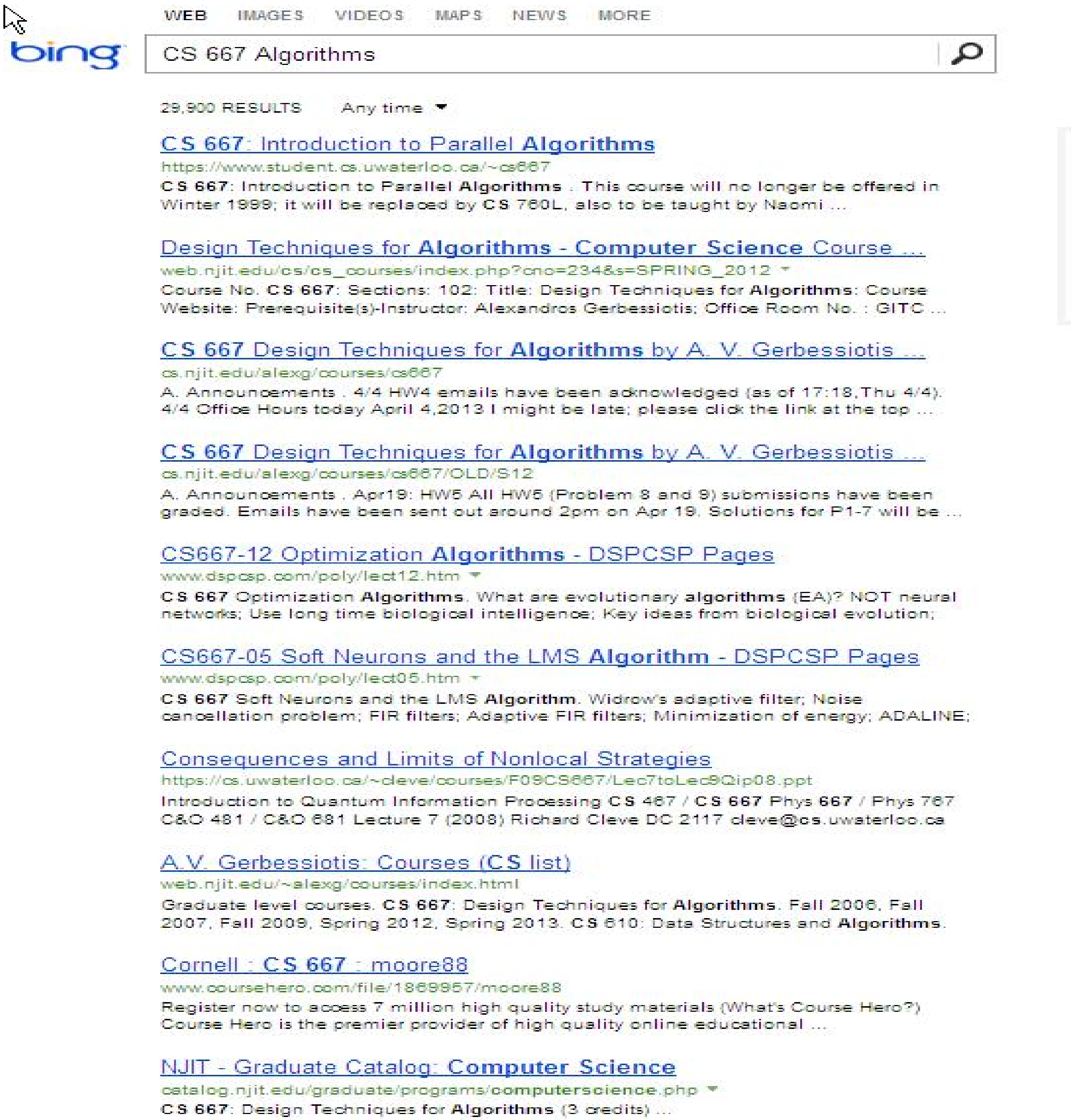
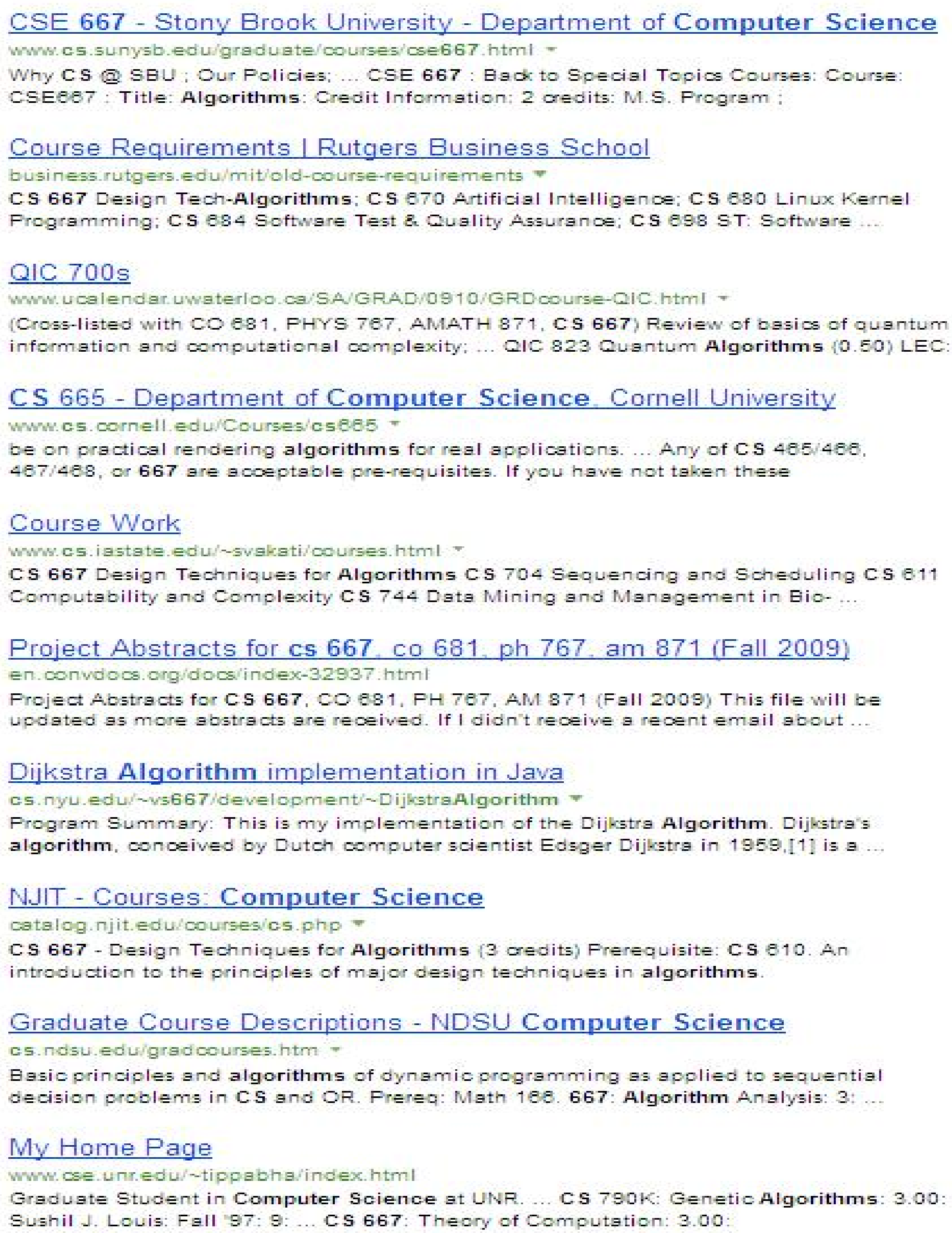


Figure 4: Bing for `CS 667 Algorithms`



CSE 667 - Stony Brook University - Department of Computer Science
www.cs.sunysb.edu/graduate/courses/cse667.html
Why CS @ SBU ; Our Policies; ... CSE 667 : Back to Special Topics Courses: Course: CSE667 : Title: Algorithms: Credit Information: 2 credits: M.S. Program ;

Course Requirements | Rutgers Business School
business.rutgers.edu/mit/old-course-requirements
CS 667 Design Tech-Algorithms; CS 670 Artificial Intelligence; CS 680 Linux Kernel Programming; CS 684 Software Test & Quality Assurance; CS 698 ST: Software ...

QIC 700s
www.ucalendar.uwaterloo.ca/SA/GRAD/0910/GRDcourse-QIC.html
(Cross-listed with CO 681, PHYS 767, AMATH 871, CS 667) Review of basics of quantum information and computational complexity; ... QIC 823 Quantum Algorithms (0.50) LEC:

CS 665 - Department of Computer Science, Cornell University
www.cs.cornell.edu/Courses/cs665
be on practical rendering algorithms for real applications. ... Any of CS 465/466, 467/468, or 667 are acceptable pre-requisites. If you have not taken these

Course Work
www.cs.iastate.edu/~svakati/courses.html
CS 667 Design Techniques for Algorithms CS 704 Sequencing and Scheduling CS 611 Computability and Complexity CS 744 Data Mining and Management in Bio- ...

Project Abstracts for cs 667, co 681, ph 767, am 871 (Fall 2009)
en.convdocs.org/docs/index-32937.html
Project Abstracts for CS 667, CO 681, PH 767, AM 871 (Fall 2009) This file will be updated as more abstracts are received. If I didn't receive a recent email about ...

Dijkstra Algorithm implementation in Java
cs.nyu.edu/~vs667/development/~DijkstraAlgorithm
Program Summary: This is my implementation of the Dijkstra Algorithm. Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959,[1] is a ...

NJIT - Courses: Computer Science
catalog.njit.edu/courses/cs.php
CS 667 - Design Techniques for Algorithms (3 credits) Prerequisite: CS 610. An introduction to the principles of major design techniques in algorithms.

Graduate Course Descriptions - NDSU Computer Science
cs.ndsu.edu/gradcourses.htm
Basic principles and algorithms of dynamic programming as applied to sequential decision problems in CS and OR. Prereq: Math 166. 667: Algorithm Analysis: 3: ...

My Home Page
www.cse.unr.edu/~tippabha/index.html
Graduate Student in Computer Science at UNR. ... CS 790K: Genetic Algorithms: 3.00: Sushil J. Louis: Fall '97: 9: ... CS 667: Theory of Computation: 3.00:

Figure 5: Bing for `CS 667 Algorithms`