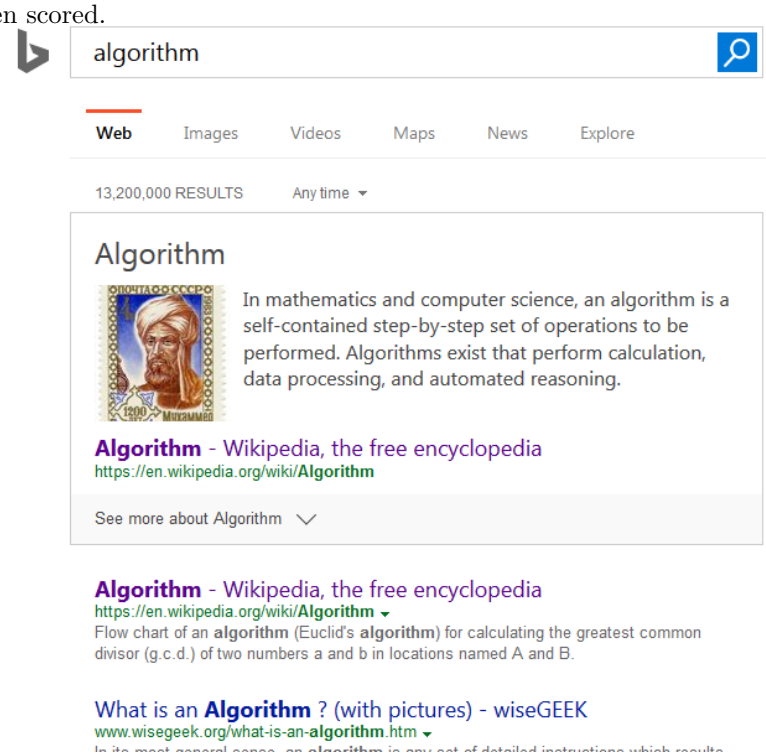
Homework 11 sample solution

Due 10/11/16

October 6, 2016

When a search engine runs a user's query, it returns the top results that it thinks the user would like to see, generally around 10 or so. The top-k search problem asks you to find the k largest values in a given unsorted array of length n, where $n \gg k$, containing real numbers representing search results that have been scored.



1. Describe an efficient algorithm to solve the top-k search problem. Acceptable algorithms might be $O(n + k \lg n)$ or $O(n \lg k)$, but not $O(nk)$ or $O(n \lg n)$.

```
   Input: result: array containing search results
   Input: n: size of result
   Input: k: number of top results to find
   Output: top k results in result
1  Algorithm: TopK
2  topk = Array()
3  heap = MaxHeap(result)
4  for i = 1 to k do
5  |  top = heap.DeleteMax()
6  |  Add top to topk
7  end
8  return topk
```

Checking for $k > n$ would also be a good idea. The following algorithm would also be acceptable:

```
   Input: result: array containing search results
   Input: n: size of result
   Input: k: number of top results to find
   Output: top k results in result
1  Algorithm: TopK2
2  heap = MinHeap()
3  for i = 1 to k do
4  |  heap.Insert(result[i])
5  end
6  for i = k + 1 to n do
7  |  if result[i] > heap.Min() then
8  |  |  heap.DeleteMin()
9  |  |  heap.Insert(result[i])
10 |  end
11 end
12 return heap
```

Alternatively, DeleteMin() and Insert() in lines 8–9 could have been replaced with IncreaseKey(), which is a *heap* method we did not discuss in class.