Heaps of Fun Might Be OK

September 24, 2016

You're managing a major online tournament of the hot new game Flappy Squirrel. There are a huge number of users, each with a competitiveness rating (a floating point number). You need an algorithm that—given a desired number of competitors c and a list of these competitiveness ratings (an array A of length n)—returns a list of the c highest ratings. You're guaranteed that $c \leq n$. (Note: we use 1-based indexing on arrays.)

1 Algorithm 1

Give and briefly justify a good asymptotic upper-bound (i.e., big-O bound) on the runtime of the following algorithm to solve this problem. (**Note:** the `buildMaxHeap` operation returns a max-heap built from the elements of a given array of length n in $O(n)$ time.)

```
TopC(A, c):
  best <- empty list
  h <- buildMaxHeap(A)
  for i = 1 to c:
    add findMax(h) to best
    deleteMax(h)
  return best
```

2 Algorithm 2

Give and briefly justify a good asymptotic upper-bound (i.e., big-O bound) on the runtime of the following algorithm to solve this problem. (Note: the notation `A[1..c]` produces a list of the elements `A[1], A[2], A[3], ..., A[c]` in $O(c)$ time.)

```
TopC(A, c):
  for i = 1 to c:
    maxIndex = i
    for j = i+1 to n:
      if A[j] > A[maxIndex]:
        maxIndex = j
    max = A[maxIndex]
    A[maxIndex] = A[i]
    A[i] = max
  return A[1..c]
```

3 Algorithm 3

Give and briefly justify a good asymptotic upper-bound (i.e., big-O bound) on the runtime of the following algorithm to solve this problem.

```
TopC(A, c):
  sort A using an efficient, comparison-based sorting algorithm
  return A[1..c]
```

4 Algorithm 4

Give and briefly justify a good asymptotic upper-bound (i.e., big-O bound) on the runtime of the following algorithm to solve this problem. (Note: `Elements(h)` produces all elements in the heap `h` in constant time, but `h` can no longer be used after that point.)

```
TopC(A, c):
  h <- empty min-heap
  for i = 1 to n:
    if Size(h) < c:
      Insert(h, A[i])
    else if A[i] > FindMin(h):
      DeleteMin(h)
      Insert(h, A[i])
  return Elements(h)
```