Assignment 5

CS834-F16: Introduction to Information Retrieval

Fall 2016

Erika Siregar

CS Department - Old Dominion University

December 17, 2016

Question 10.3

`Compute five iterations of HITS (see Algorithm 3) and PageRank (see Figure 4.11) on the graph in Figure 10.3. Discuss how the PageRank scores compare to the hub and authority scores produced by HITS.`

Answer

Figure 1 shows the directed graph from the textbook [1] on which we will calculate the scores of HITS and PageRank. Computing HITS (authorities and hubs) and PageRank scores are pretty easy since we can just utilize the Link Analysis procedure that is provided by python library 'networkx' [2].

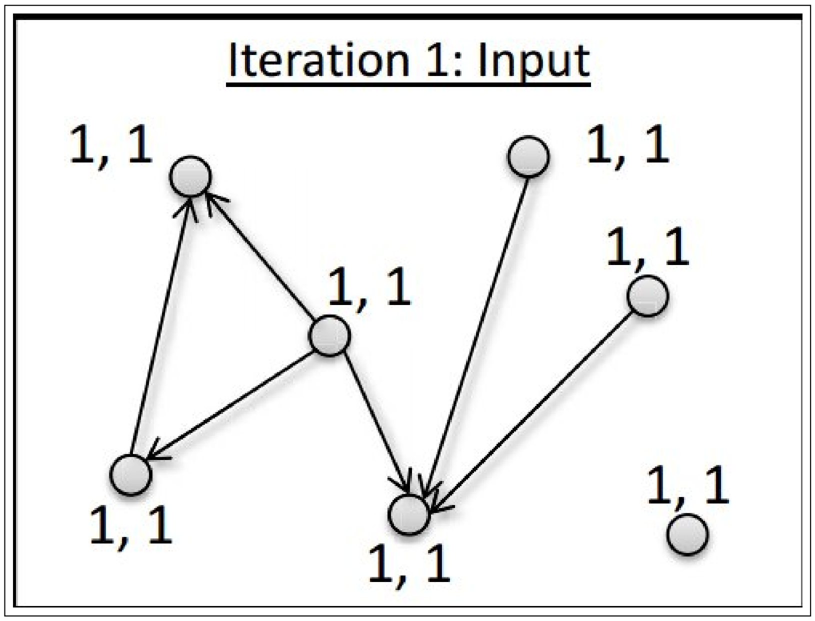
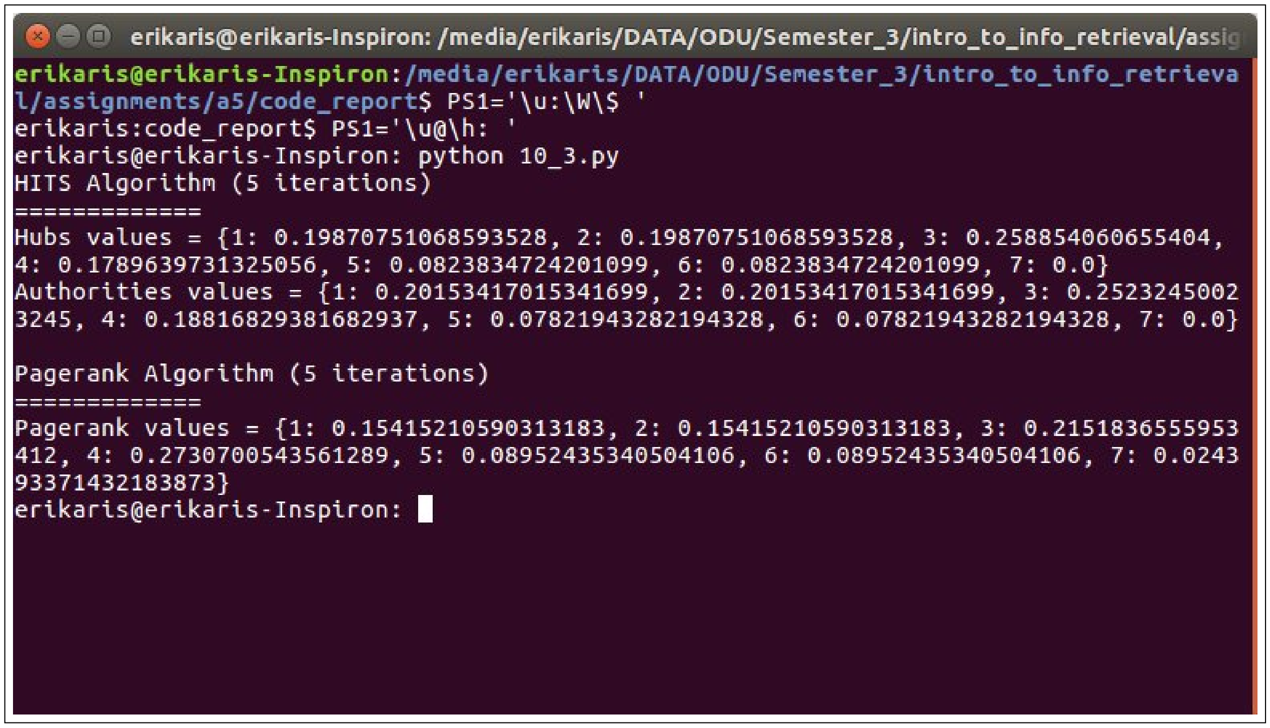


Figure 1: Figure 10.3 from the textbook [1]

Figure 2 shows the scores of HITS (authorities and hubs) and PageRank, which are obtained by running the code in listing 1. We only need to set the number of iterations.


erikaris@erikaris-Inspiron: /media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieval/assig

```
erikaris@erikaris-Inspiron:/media/erikaris/DATA/ODU/Semester_3/intro_to_info_retrieva
l/assignments/a5/code_report$ PS1='\u:\W\$ '
erikaris:code_report$ PS1='\u@\h: '
erikaris@erikaris-Inspiron: python 10_3.py
HITS Algorithm (5 iterations)
=============
Hubs values = {1: 0.19870751068593528, 2: 0.19870751068593528, 3: 0.258854060655404,
4: 0.1789639731325056, 5: 0.0823834724201099, 6: 0.0823834724201099, 7: 0.0}
Authorities values = {1: 0.20153417015341699, 2: 0.20153417015341699, 3: 0.2523245002
3245, 4: 0.18816829381682937, 5: 0.07821943282194328, 6: 0.07821943282194328, 7: 0.0}

Pagerank Algorithm (5 iterations)
=============
Pagerank values = {1: 0.15415210590313183, 2: 0.15415210590313183, 3: 0.2151836555953
412, 4: 0.2730700543561289, 5: 0.08952435340504106, 6: 0.08952435340504106, 7: 0.0243
93371432183873}
erikaris@erikaris-Inspiron: 
```

Figure 2: HITS and Pagerank for Figure 10.3 with 5 Iterations

To make the analysis and comparison easier, I transformed the output int figure 2 into a neat table format as can be seen on table 1. From table 1, we can see that, generally, the authorities values are linearly proportional to those of PageRank. After 5 iterations, node 3 gets the highest score for 'authorities' and the second highest score for 'PageRank'. Nodes 1 and 2 get lower 'authorities' score than that of node 3, but higher 'authorities' score compare to nodes 5 and 6. The same thing can also be concluded by comparing the PageRank scores for those five nodes (1, 2, 3, 5, and 6). The strange thing happens on node 4, where its 'authorities' score is lower than node 3, but its 'PageRank' score is higher than node 3. This anomaly takes place probably because we only do 5 iterations. Maybe, if we continue iterating until the values converge into certain number, this anomaly will not happen.

Node	**Score**		
	Hubs	**Authorities**	**PageRank**
1	0.198707510685935	0.201534170153416	0.154152105903131
2	0.198707510685935	0.201534170153416	0.154152105903131
3	0.258854060655404	0.252324500232450	0.215183655595341
4	0.178963973132505	0.188168293816829	0.273070054356128
5	0.082383472420110	0.078219432821943	0.089524353405041
6	0.082383472420110	0.078219432821943	0.089524353405041
7	0.000000000000000	0.000000000000000	0.024393371432184

Table 1: HITS and Pagerank for Figure 10.3 with 5 Iterations

```python
#!/usr/bin/python

import networkx as nx

```

```
def hits(G, iter=100, nstart=None, normalized=True):
if type(G) == nx.MultiGraph or type(G) == nx.MultiDiGraph:
raise Exception("hits() not defined for graphs with multiedges.")
if len(G) == 0:
return {},{}
# choose fixed starting vector if not given
if nstart is None:
h=dict.fromkeys(G,1.0/G.number_of_nodes())
else:
h=nstart
# normalize starting vector
s=1.0/sum(h.values())
for k in h:
h[k]*=s
i=0
while True: # power iteration: make up to max_iter iterations
if i >= iter: break

hlast=h
h=dict.fromkeys(hlast.keys(),0)
a=dict.fromkeys(hlast.keys(),0)
# this "matrix multiply" looks odd because it is
# doing a left multiply a^T=hlast^T*G
for n in h:
for nbr in G[n]:
a[nbr]+=hlast[n]*G[n][nbr].get('weight',1)
# now multiply h=Ga
for n in h:
for nbr in G[n]:
h[n]+=a[nbr]*G[n][nbr].get('weight',1)
# normalize vector
s=1.0/max(h.values())
for n in h: h[n]*=s
# normalize vector
s=1.0/max(a.values())
for n in a: a[n]*=s

i+=1
if normalized:
s = 1.0/sum(a.values())
for n in a:
a[n] *= s
s = 1.0/sum(h.values())
for n in h:
h[n] *= s
return h,a

def pagerank(G, alpha=0.85, personalization=None,
iter=100, nstart=None, weight='weight',
dangling=None):
if len(G) == 0:
return {}

if not G.is_directed():
D = G.to_directed()
else:
D = G

# Create a copy in (right) stochastic form
```

```
W = nx.stochastic_graph(D, weight=weight)
N = W.number_of_nodes()

# Choose fixed starting vector if not given
if nstart is None:
x = dict.fromkeys(W, 1.0 / N)
else:
# Normalized nstart vector
s = float(sum(nstart.values()))
x = dict((k, v / s) for k, v in nstart.items())

if personalization is None:
# Assign uniform personalization vector if not given
p = dict.fromkeys(W, 1.0 / N)
else:
missing = set(G) - set(personalization)
if missing:
raise nx.NetworkXError('Personalization dictionary '
'must have a value for every node. '
'Missing nodes %s' % missing)
s = float(sum(personalization.values()))
p = dict((k, v / s) for k, v in personalization.items())

if dangling is None:
# Use personalization vector if dangling vector not specified
dangling_weights = p
else:
missing = set(G) - set(dangling)
if missing:
raise nx.NetworkXError('Dangling node dictionary '
'must have a value for every node. '
'Missing nodes %s' % missing)
s = float(sum(dangling.values()))
dangling_weights = dict((k, v/s) for k, v in dangling.items())
dangling_nodes = [n for n in W if W.out_degree(n, weight=weight) == 0.0]

# power iteration: make up to max_iter iterations
for _ in range(iter):
xlast = x
x = dict.fromkeys(xlast.keys(), 0)
danglesum = alpha * sum(xlast[n] for n in dangling_nodes)
for n in x:
# this matrix multiply looks odd because it is
# doing a left multiply x^T=xlast^T*W
for nbr in W[n]:
x[nbr] += alpha * xlast[n] * W[n][nbr][weight]
x[n] += danglesum * dangling_weights[n] + (1.0 - alpha) * p[n]

return x

if __name__ == '__main__':
iter = 5
G = nx.Graph()

# Add 7 nodes
G.add_nodes_from(range(1,8))

# Add 6 edges
G.add_edges_from([(1,2), (3,1), (3,2), (3,4), (5,4), (6,4)])
```

```

# Compute hubs and authorities normalized values using hits
h, a = hits(G, iter=iter)

print 'HITS Algorithm ({} iterations)'.format(iter)
print '================'
print 'Hubs values = {}'.format(h)
print 'Authorities values = {}'.format(a)
print ''

# Compute pagerank of each nodes
pr = pagerank(G, iter=iter)

print 'Pagerank Algorithm ({} iterations)'.format(iter)
print '================'
print 'Pagerank values = {}'.format(pr)
```

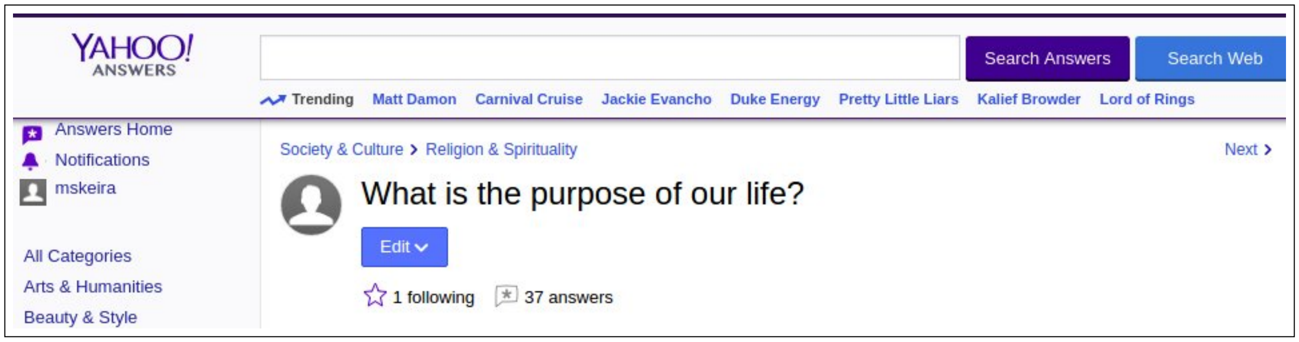
Listing 1: Computing HITS and PageRank

Question 10.5

`Find a community-based question answering site on the Web and ask two questions, one that is low-quality and one that is high-quality. Describe the answer quality of each question.`

Answer:

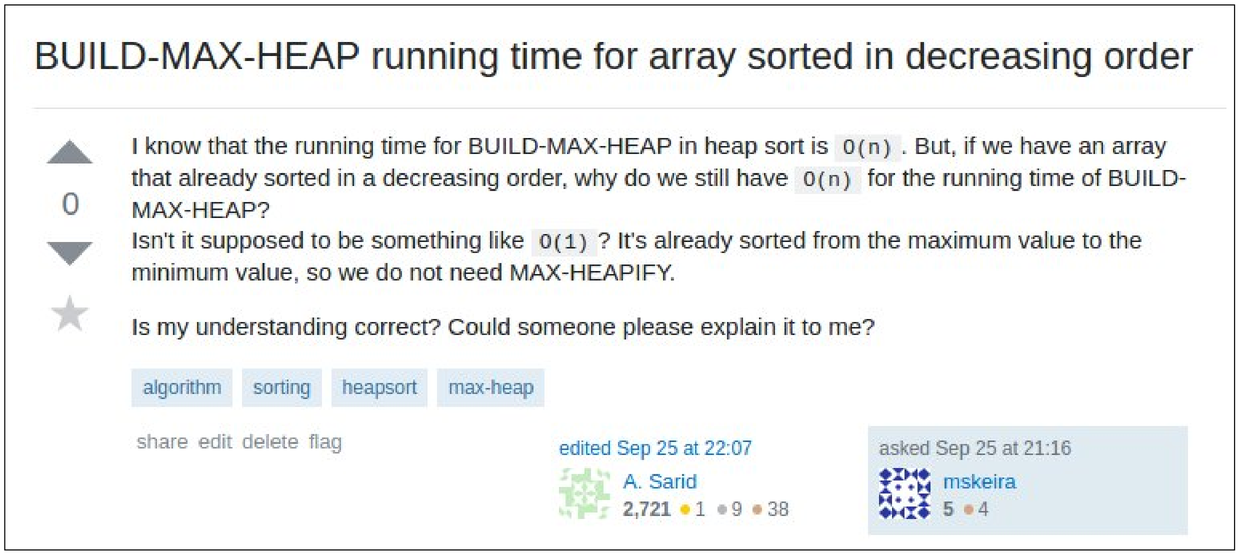
For this assignment, I asked 2 questions on 2 different comunity-based question answering site. For the low-quality question, I asked about ***'What is the purpose of our life?'*** [1] on Yahoo Answers `https://answers.yahoo.com/` as can be seen on figure 3. For the high-quality question, I asked the question ***'BUILD-MAX-HEAP running time for array sorted in decreasing order'*** on Stackoverflow [2] as can be seen on figure 5.



What is the purpose of our life?

Figure 3: Low Quality Question I asked on Yahoo Answers

[1] https://answers.yahoo.com/question/index?qid=20161216122515AAPvTIwpage=4
[2] http://stackoverflow.com/questions/39691923/build-max-heap-running-time-for-array-sorted-in-decreasing-order



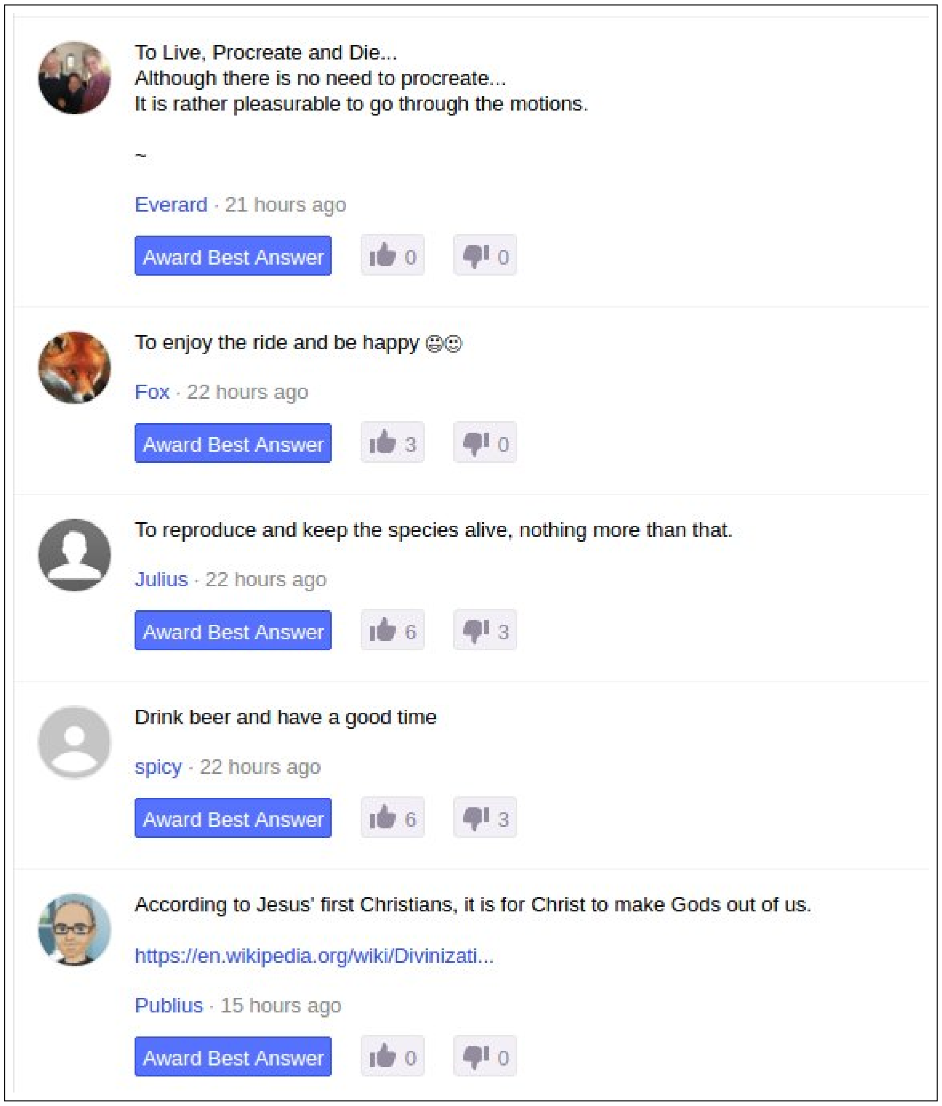
BUILD-MAX-HEAP running time for array sorted in decreasing order

I know that the running time for BUILD-MAX-HEAP in heap sort is `O(n)`. But, if we have an array that already sorted in a decreasing order, why do we still have `O(n)` for the running time of BUILD-MAX-HEAP?
Isn't it supposed to be something like `O(1)`? It's already sorted from the maximum value to the minimum value, so we do not need MAX-HEAPIFY.

Is my understanding correct? Could someone please explain it to me?

Figure 4: High Quality Question I asked on Stackoverflow

By the time I write this report, I got 37 answers for the low-quality question on Yahoo Answers. Some of the answers can be seen on figure 5. For the high-quality question on Stackoverflow, I only got 2 answers as can be seen on figure 6.
However, for the low-quality question, I also got low-quality answers. This is understandable because for a low-quality question, people tend to post anything that they have in their minds without being afraid of any risks. For example, when I asked 'What is the purpose of our life', I got answers like 'Who says that there's one?', 'To study the paintings of the great Masters, and understand that UFOs are real', and 'Drink beer and have a good time'.
Accordingly, despite the low number of answers, I got high-quality answers for the high-quality question. This is understandable because for this type of question, people will think twice (or maybe more) before submitting the answers. They should be able to provide not only answer, but also the explanation why the answer is correct. People will not take the risk of embarassing themselves by saying something irrelevant. For a high-quality answer, not everyone has the ability to provide a justifiable answer and explanation. Hence, we got a lower number of answers for the high-quality question compare to the low-quality question.


To Live, Procreate and Die...
Although there is no need to procreate...
It is rather pleasurable to go through the motions.

~

Everard · 21 hours ago

Award Best Answer 0 0

To enjoy the ride and be happy 😄🙂

Fox · 22 hours ago

Award Best Answer 3 0

To reproduce and keep the species alive, nothing more than that.

Julius · 22 hours ago

Award Best Answer 6 3

Drink beer and have a good time

spicy · 22 hours ago

Award Best Answer 6 3

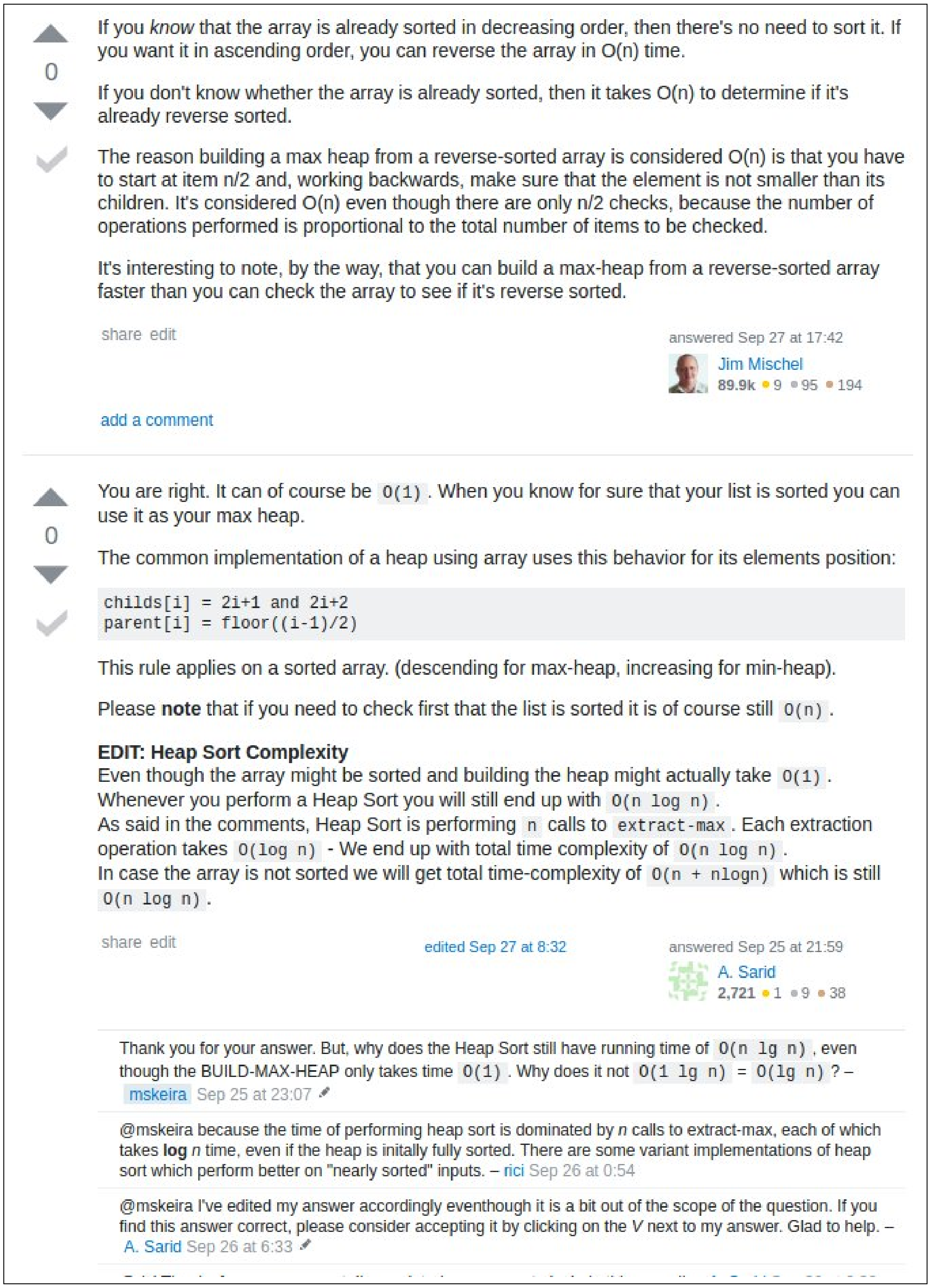
According to Jesus' first Christians, it is for Christ to make Gods out of us.

https://en.wikipedia.org/wiki/Divinizati...

Publius · 15 hours ago

Award Best Answer 0 0

Figure 5: Answers for the low-quality question I asked on Yahoo Answers



0

If you *know* that the array is already sorted in decreasing order, then there's no need to sort it. If you want it in ascending order, you can reverse the array in O(n) time.

If you don't know whether the array is already sorted, then it takes O(n) to determine if it's already reverse sorted.

The reason building a max heap from a reverse-sorted array is considered O(n) is that you have to start at item n/2 and, working backwards, make sure that the element is not smaller than its children. It's considered O(n) even though there are only n/2 checks, because the number of operations performed is proportional to the total number of items to be checked.

It's interesting to note, by the way, that you can build a max-heap from a reverse-sorted array faster than you can check the array to see if it's reverse sorted.

share edit

answered Sep 27 at 17:42
Jim Mischel
89.9k ● 9 ● 95 ● 194

add a comment

0

You are right. It can of course be `O(1)`. When you know for sure that your list is sorted you can use it as your max heap.

The common implementation of a heap using array uses this behavior for its elements position:

```
childs[i] = 2i+1 and 2i+2
parent[i] = floor((i-1)/2)
```

This rule applies on a sorted array. (descending for max-heap, increasing for min-heap).

Please **note** that if you need to check first that the list is sorted it is of course still `O(n)`.

EDIT: Heap Sort Complexity
Even though the array might be sorted and building the heap might actually take `O(1)`.
Whenever you perform a Heap Sort you will still end up with `O(n log n)`.
As said in the comments, Heap Sort is performing `n` calls to `extract-max`. Each extraction operation takes `O(log n)` - We end up with total time complexity of `O(n log n)`.
In case the array is not sorted we will get total time-complexity of `O(n + nlogn)` which is still `O(n log n)`.

share edit

edited Sep 27 at 8:32

answered Sep 25 at 21:59
A. Sarid
2,721 ● 1 ● 9 ● 38

Thank you for your answer. But, why does the Heap Sort still have running time of `O(n lg n)`, even though the BUILD-MAX-HEAP only takes time `O(1)`. Why does it not `O(1 lg n)` = `O(lg n)`? – mskeira Sep 25 at 23:07

@mskeira because the time of performing heap sort is dominated by *n* calls to extract-max, each of which takes **log** *n* time, even if the heap is initally fully sorted. There are some variant implementations of heap sort which perform better on "nearly sorted" inputs. – rici Sep 26 at 0:54

@mskeira I've edited my answer accordingly eventhough it is a bit out of the scope of the question. If you find this answer correct, please consider accepting it by clicking on the *V* next to my answer. Glad to help. – A. Sarid Sep 26 at 6:33

Figure 6: Answers for the high-quality question I asked on Stackoverflow

Question 8.5

`question 3`

Answer

answer 3

Question 8.7

`question 4`

Answer

answer 4

References

[1] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 1st edition, 2009.

[2] NetworkX Developers. Networkx - Link Analysis. `https://networkx.github.io/documentation/networkx-1.9/reference/algorithms.link_analysis.html`, 2016. [Online; accessed 14-December-2016].