# CS317: Algorithms — Homework Assignment #6

Due: See Canvas for Assignment Due Dates (20 points)

*NOTE: NO LATE ASSIGNMENTS WILL BE ACCEPTED because we often review them in class on the due date.*

Please upload the document containing your answers. They can be handwritten and scanned, but they must be clearly legible to receive a grade on the assignment. PDF is the best format for canvas. You DO NOT Need to include this cover sheet in your upload. It is formatted for the grader to use for me, as needed.

Chapter 6: Work the following problems. Point values are provided for each problem.

| Problem # | Points | Grader's Notes |
|---|---|---|
| Sec 6.4, **#1b,c** | *5* | |
| Sec 6.4, #5a | *2* | *Describe the approach, you don't have to write a complete pseudocode* |
| Sec 6.4, #7 | *3* | |
| Sec 6.4, #8 | *1* | |

Chapter 7: Work the following problems. Point values are provided for each problem.

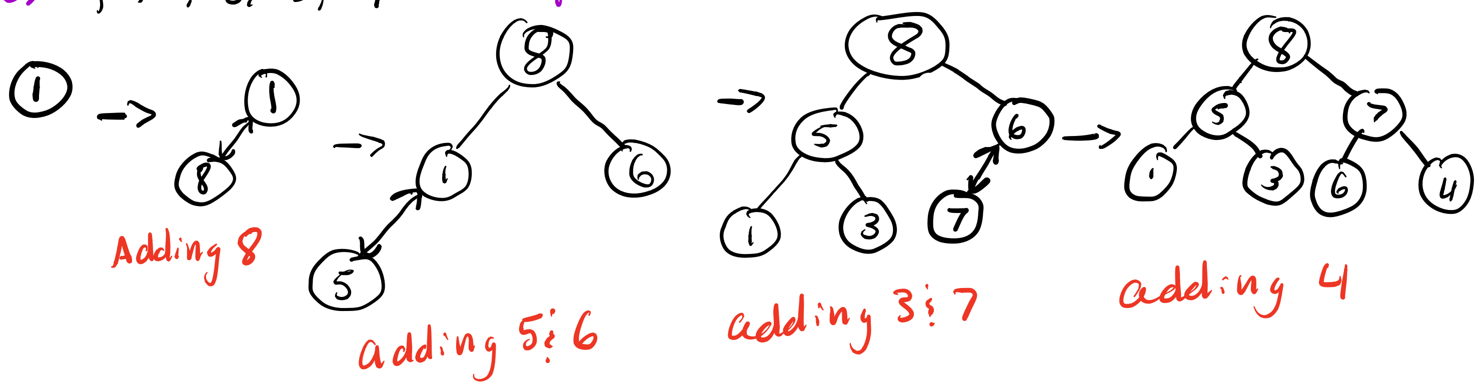
| Problem # | Points | Grader's Notes |
|---|---|---|
| Sec 7.1, #1 | *2* | *Explain how* |
| Sec 7.1, #2 | *2* | |
| Sec 7.1, #3 | *2* | |
| Sec 7.1, #4 | *1* | |
| Sec 7.1, #5 | *2* | |

**1. a.** Construct a heap for the list 1, 8, 6, 5, 3, 7, 4 by the bottom-up algorithm.

**b.** Construct a heap for the list 1, 8, 6, 5, 3, 7, 4 by successive key insertions (top-down algorithm).

**c.** Is it always true that the bottom-up and top-down algorithms yield the same heap for the same input?

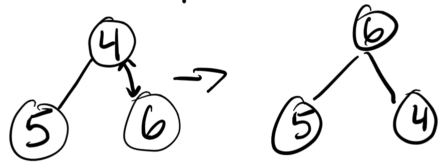
b) 1, 8, 6, 5, 3, 7, 4 (top down)



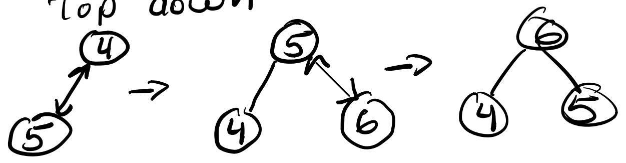
c) Not always. Since bottom up goes from bottom to top, it is comparing the child up to parent whereas top down compares parent down to child. Therefore, you're the swaps of parent and child may differ.

ex:

Bottom up



Top down



---

**5. a.** Design an efficient algorithm for finding and deleting an element of the smallest value in a heap and determine its time efficiency.

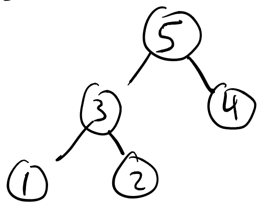
```
Algorithm   Smallest Leaf( H[1...n])
    // Scans second half of the array
    while not at end                    i gets first value in second half.
        i ← H[n/2+1]
        if i < H(N)                     if i is less than last value
            swap i & H(n)               move H(n) -> i's pos.
            delete i.                   // remove I
        else i = i+1                    // else go to next element
    return.                             to compare
```

should not need this since should be sorted.

**7.** Sort the following lists by heapsort by using the array representation of heaps.
**a.** 1, 2, 3, 4, 5 (in increasing order)
**b.** 5, 4, 3, 2, 1 (in increasing order)
**c.** S, O, R, T, I, N, G (in alphabetical order)

a)

```
1   2   3   4   5
1   5   3   4   2
1   5   4   3   2
5   1   4   3   2
5   3   4   1   2
```



```
5   3   4   1   2
2   3   4   1  [5   delete.
4   3   2   1
1   3   2  [4   delete.
3   1   2
1   2  [3   delete
1  [2   delete
1
```

b)

5, $\overset{\downarrow}{4}$, 3, 2, 1

$\overset{\downarrow}{5}$, 4, 3, 2, 1

```
5  4  3  2  1
1  4  3  2 | 5
4  2  3  1
1  2  3 | 4
3  2  1
1  2 | 3
2  1
1 | 2
1
```

$\underline{5}$ $\underline{4}$ $\underline{3}$ $\underline{2}$ $\underline{1}$

A

c)

```
S  ↓O  R   T   I   N   G
S   T  ↓R  O   I   N   G
S   T   O  R  ↓I   N   G
S   T   O  R   G  ↓N   I
↓S  T   O  R   G   I   N
T  ↓S   O  R   G   I   N
T  ↓O   S  R   G   I   N
T   R   S  O   G   I   N
```

```
T  R  S  O  G  I  N
N  R  S  O  G  I | (T) - delete
S  R  N  O  G  I
I  R  N  O  G | (S) - delete
R  I  N  O  G
G  I  N  O | (R) - delete
O  I  N  G
G  I  N | (O) - delete
N  I  G
G  I | (N) - delete
I  G
G | (I) - delete
(G)
```

**8.** Is heapsort a stable sorting algorithm?

No because sometimes a swap may change the original placing of the elements

---

# Chapter 7

**1.** Is it possible to exchange numeric values of two variables, say, *u* and *v*, without using any extra storage?

You could swap using a temporary variable, but that takes space...

```
u = u+v    // update u     6 = 2 +4
v = u-v    // v gets u     2 = 6 - 4
u = u-v    // u gets v     4 = 6 - 2
```

**2.** Will the comparison-counting algorithm work correctly for arrays with equal values?

It should, yes since it would just be populated 1, 2, 3, 4, .....

you would get the same array back

**3.** Assuming that the set of possible list values is {*a, b, c, d*}, sort the following list in alphabetical order by the distribution-counting algorithm:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *b,* | *c,* | *d* | *c* | *b,* | *a,* | *a* | *b.* |
| 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 |

Frequencies:

| a | b | c | d |
|---|---|---|---|
| 2 | 3 | 2 | 1 |

Distribution

| a | b | c | d |
|---|---|---|---|
| 2 | 5 | 7 | 8 |

| | d[a..d] | | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A[7] = b | 2 | 5 | 7 | 8 | — | | | | | b | | | |
| A[6] = a | 2 | 4 | 7 | 8 | — | | a | | | | | | |
| A[5] = a | 1 | 4 | 7 | 8 | — | a | | | | | | | |
| A[4] = b | 0 | 4 | 7 | 8 | — | | | | b | | | | |
| A[3] = c | 0 | 3 | 7 | 8 | — | | | | | | | c | |
| A[2] = d | 0 | 3 | 6 | 8 | — | | | | | | | | d |
| A[1] = c | 0 | 3 | 6 | 7 | — | | | | | | c | | |
| A[0] = b | 0 | 3 | 5 | 7 | — | | | b | | | | | |

S[0...7]

---

**4.** Is the distribution-counting algorithm stable?

Yes because it scans Right to left and inserts right to left as well.

---

**5.** Design a one-line algorithm for sorting any array of size *n* whose values are *n* distinct integers from 1 to *n*.

for i ←0 to n-1 do S[A[i] -1] ←A[i]

↑ element 0

↑ last element

↑ previous index gets the current index's element.