Written Homework 3

Due 23:59, Thursday, June 21, 2018

CS ID 1:	d9p1b
CS ID 2:	

Each individual or pair will be submit one completed copy of this homework onto gradescope. You may either print it and write your solutions in the space provided, or complete the assignment in latex. Show your work where necessary for full credit. If you require additional space, please indicate in the question space that you are writing on the last blank page, and also indicate on the blank page which question the work solves.

You must upload the completed document, including this page and the last page, to GradeScope, using your 4- or 5-character CS ID.

1. **[Multiple Choice — 30 points].**

   In each of the multiple choice problems below, bubble the best answer. In a few cases, more than one response is correct. Mark them all! A good way to solve these problems is to speculate on a response *without* using the compiler, and then check to see if the code behaves as you predicted.

Sorting (6 points)

Suppose we have an array of integers entered as follows: 5, 8, 7, 1, 6, 3, 9, 0.
Using Jon Bentley's Quicksort to split the array into two partitions, which of the following intermediary arrays are possible during the *first* iteration, when `5` is the pivot.

☐ {5, 0, 7, 1, 6, 3, 9, 8}
■ {5, 1, 7, 8, 6, 3, 9, 0}
☐ {5, 1, 3, 0, 8, 7, 6, 9}
■ {5, 1, 3, 8, 6, 7, 9, 0}
■ {0, 1, 3, 5, 6, 7, 9, 8}
☐ {0, 1, 3, 5, 6, 7, 8, 9}
☐ None of the above.

Suppose we have an array of integers entered as follows: 5, 8, 7, 1, 6, 3, 9, 0.
Using Insertion sort, which of the following intermediary arrays are possible?

■ {5, 8, 7, 1, 6, 3, 9, 0}
☐ {5, 1, 8, 7, 6, 3, 9, 0}
☐ {5, 1, 3, 8, 7, 6, 9, 0}
■ {5, 7, 8, 1, 6, 3, 9, 0}
■ {1, 5, 7, 8, 6, 3, 9, 0}
■ {1, 3, 5, 6, 7, 8, 9, 0}
☐ None of the above.

Suppose we have an array of integers entered as follows: 5, 8, 7, 1, 6, 3, 9, 0.
Using Selection sort, which of the following intermediary arrays are possible?

■ {0, 8, 7, 1, 6, 3, 9, 5}
☐ {1, 8, 7, 5, 6, 3, 9, 0}
☐ {0, 1, 5, 8, 7, 6, 3, 9}
■ {0, 1, 3, 8, 6, 7, 9, 5}
■ {0, 1, 3, 5, 6, 7, 9, 8}
☐ {0, 1, 3, 5, 6, 8, 9, 7}
☐ None of the above.

Hashing (6 points)

Given an empty hash table with a capacity for 11 elements, what are the contents of the hash table after inserting the following items in the order given: 43, 25, 14, 91, 07, 33, 58, 17, 03.

Assuming the collision resolution strategy is linear probing. For collisions, attempt to find a new location m times.

○	33	58	03	25	00	14	17	91	00	07	43
○	33	00	00	25	14	00	00	91	07	00	43
●	33	00	00	25	14	91	58	07	17	03	43
○	33	00	03	25	14	58	91	07	00	17	43
○	33	00	03	25	14	91	17	07	58	00	43
○	33	58	00	25	14	00	17	91	07	00	43

○ None of the above

Assuming the collision resolution strategy is quadratic probing. For collisions, attempt to find a new location m times.

○	33	58	03	25	00	14	17	91	00	07	43
○	33	00	00	25	14	00	00	91	07	00	43
○	33	00	00	25	14	91	58	07	17	03	43
○	33	00	03	25	14	58	91	07	00	17	43
○	33	00	03	25	14	91	17	07	58	00	43
●	33	58	00	25	14	00	17	91	07	00	43

○ None of the above

Assuming the collision resolution strategy is double hashing. For collisions, attempt to find a new location m times. Assume $h2(key) = 5 - key\%5$

○	33	58	17	25	03	00	14	07	00	91	43
○	33	58	03	25	00	14	17	91	00	07	43
○	33	00	07	25	03	14	17	91	00	58	43
●	33	00	07	25	14	58	17	91	00	03	43
○	33	03	00	25	91	00	58	14	17	07	43
○	33	03	00	25	14	00	17	91	58	07	43

○ None of the above

2. **[Loop Invariant Proof – 5 points].**

   Prove the correctness of the following function, using induction. Note the loop invariant.

```
// Precondition: the size of array A is greater than 0
// Postcondition: the maximum value is returned, and the array has not been
   modified
int maximumValue (int A[], int size) {
  int largest = A[0];
  int i = 1;
  // invariant: largest = maximum in A[0..i]
  while (i <= size-1) {
    if ( A[i] > largest ) {
      largest = A[i];
    }
    i = i + 1;
  }
  return largest;
}
```

Let n be the size of the array, $n \geq 1$ (Precondition)

Base case:

If $n = 1$, `A[0]` is the only element in the list, hence the largest element in the list.

Induction hypothesis:

At iteration k, where $k < size$, we assume `largest` is the largest number in `A[0...k-1]`

Inductive step:

We want to prove that after the k^{th} iteration, `largest` is the largest number in `A[0...k]`

Inner loop gets `A[k]` and compare it with `largest`.
If `A[k] is the largest number in \lstinline{A[0...k]`, it should be greater than the largest number in `A[0...k-1]`, `largest`. Namely, `A[k] > largest` is `true`, and `largest` is updated to the new largest number `A[k]`.
Alternatively, if `largest` is the largest number in `A[0...k]`, `A[k]` should be less than or equal to the largest number in `A[0...k-1]`. We don't enter the loop and `largest` remains unchanged.

Termination:

Loop ends when `i == size`
This will happen since size is non-negative and i increases (`i = i + 1`)
By the loop invariant, `largest` is the largest number in `A[0...i-1]`, and since `i == size`, `arr[0...i-1]` is `arr[0?size-1]` which is the whole array

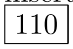
3. **[B+ Trees – 12 points].**

Insertion (6 points)

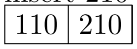
Suppose we currently have an empty B+ tree root node that can hold up to 2 keys (and the same is true for all other nodes in the tree). We want to enter the following keys, in this order, into the B+ tree: 110, 210, 121, 221, 213, 304, 320, 310

Draw the tree after each insertion; clearly show any time that a node splits.

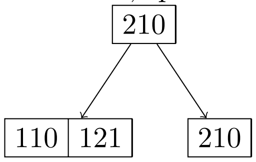
insert 110



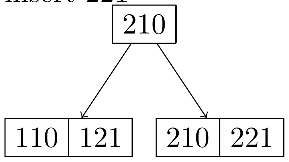
insert 210



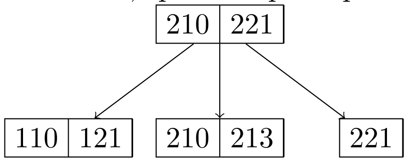
insert 121, split and pull up 210



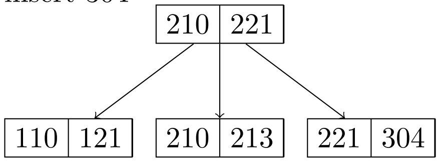
insert 221



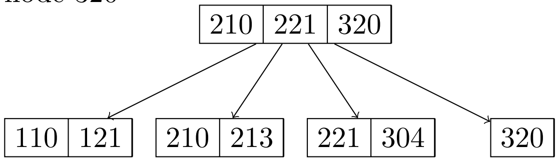
insert 213, split and pull up 221



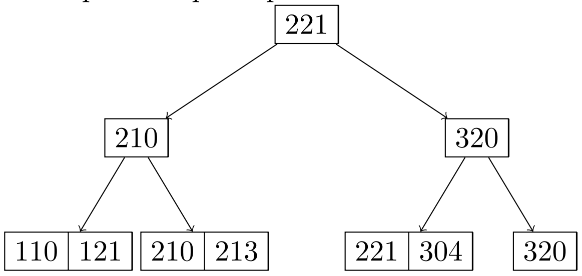
insert 304



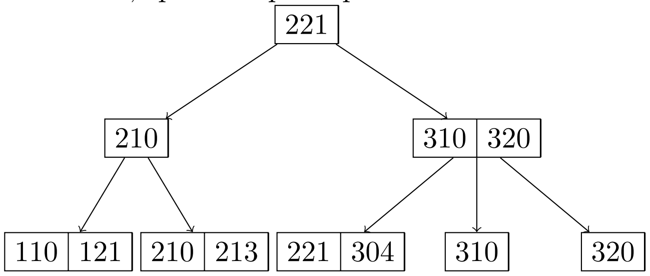
insert 320, first split and pull up the leaf node 320



then split and pull up the internal node 221



insert 310, split and pull up 310



Theory (6 points)

Suppose we want to build a B+ tree that has space for 200,000 data entries in its leaf pages. Each data entry is made up of a key and its corresponding data value. Let us assume the following specifications. Each page (leaf or internal) is 4096 bytes long. Each page holds three 8-byte pointers (parent, left sibling, right sibling) in addition to the bytes consumed by all of the keys and their accompanying 8-byte values. (In the case of leaf pages, these values are simply pointers that locate the full data record corresponding to the key (e.g., customer record for the given key.) We want to use an even number of data entries in the leaf pages. Internally, we also want an even number. The keys have unique values. For all of the following questions, show your work:

(a) Suppose each key is 56 bytes long, and suppose we fill the leaf pages to capacity (i.e., as much as possible). Compute the number of leaf pages that we need.
For each page, the space to store key value pairs is

$$4096 - 3 \times 8 = 4072 bytes$$

The number of key value pairs that a page can store is (round down to the closest even number)

$$4072/(56 + 8) = 62$$

The total leaf pages needed (round up)

$$200,000/62 = 3226$$

(b) Compute the number of internal pages, at each level, that we "need" (i.e., assume that you can fill the parents to capacity (to the maximum even number of keys)). This will result in a structure that has the fewest number of pages. Note, however, that the root can have as few as one key.
For an internal node pages, 4072 bytes space can be used to store keys and pointers. 62 keys and 63 pointers will takes up $62 \times (56 + 8) + 8 = 3976$ bytes of space. Since we want even number of data entries in internal nodes, the $4072 - 3976 = 96$ bytes left are not enough for two keys and pointers. So the maximum number of keys and pointers that we can have in an internal page are 62 and 63. In other words, an internal node page can point to at most 63 leaf nodes.
The number of internal pages in the lowest level of the tree is (round up)

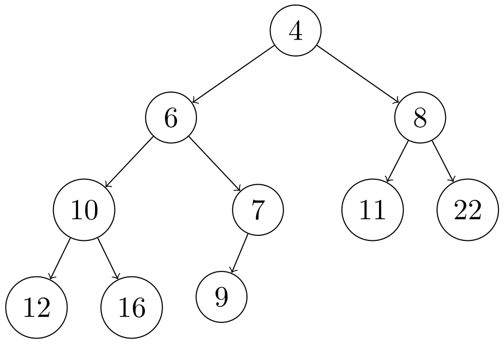
$$3226/63 = 53$$

node pages. Hence we know we will need at least 53 internal node pages.

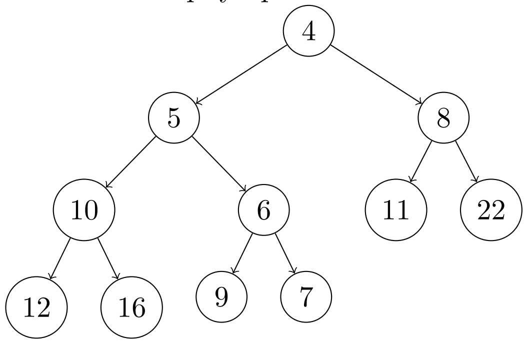
4. **[Priority Queues – 8 points].**

Insertion (4 points)

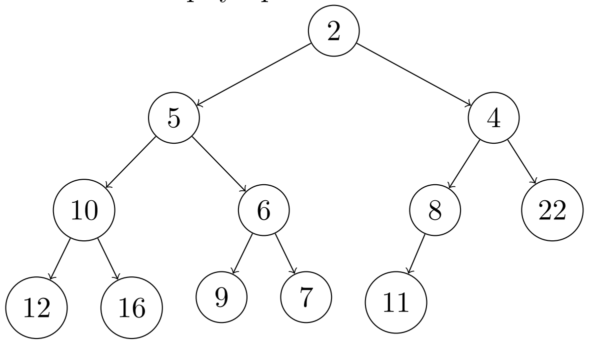
Given the following binary heap, what are the contents of the underlying array after inserting 5, 2, 27, and 14? You do not need to show all your work, but it may help you receive part marks if you final array in incorrect.



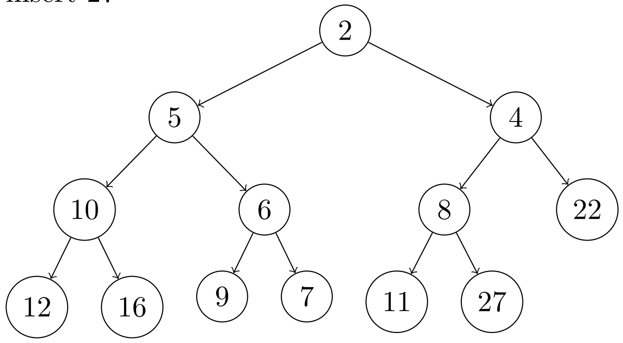
insert and heapify up 5



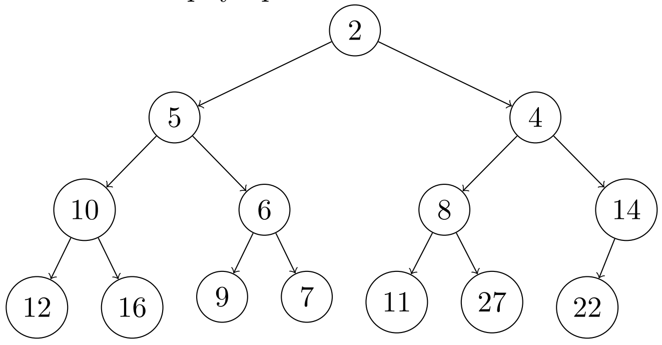
insert and heapify up 2



insert 27



insert and heapify up 14

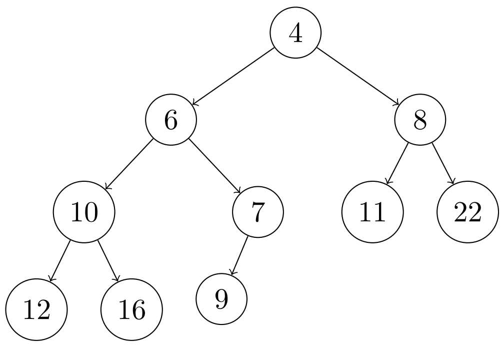


Array values:

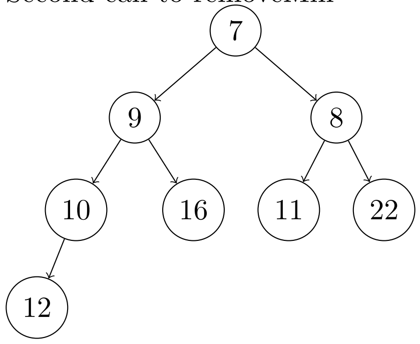
2	5	4	10	6	8	14	12	16	9	7	11	27	22

RemoveMin (4 points)

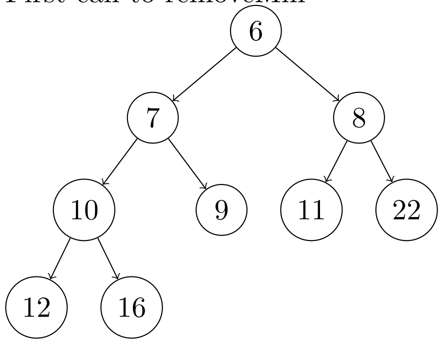
Given the following binary heap, what are the contents of the underlying array after 3 calls to removeMin? You do not need to show all your work, but it may help you receive part marks if you final array in incorrect.



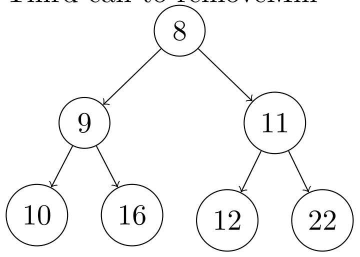
Second call to removeMin



First call to removeMin



Third call to removeMin



Array values:

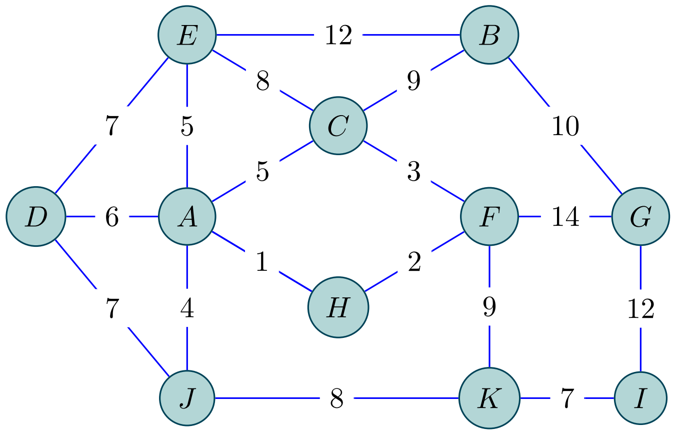
8	9	11	10	16	12	22

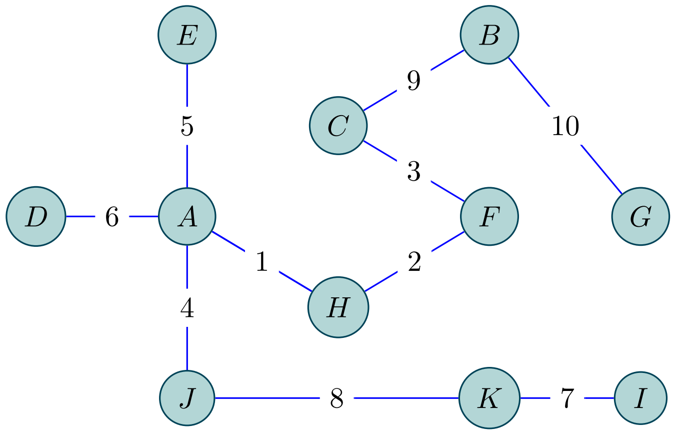
5. **[Priority Queues – 8 points].**

Kruskal's Algorithm (4 points)

Given the following graph, complete the table showing the result of Kruskal's algorithm. For each edge, indicate whether it is in the minimum spanning tree (MST) found by selecting the corresponding box. For edges that **ARE** in the resulting MST, state the order the algorithm checks it (i.e., 1, 2, 3, etc). I have begun filling the table for you.

edge	in MST?	order
AC	○	
AD	●	6
AE	●	5
AH	●	1
AJ	●	4
BC	●	9
BE	○	
BG	●	10
CE	○	
CF	●	3
DE	○	
DJ	○	
FH	●	2
FG	○	
FK	○	
GI	○	
IK	●	7
JK	●	8

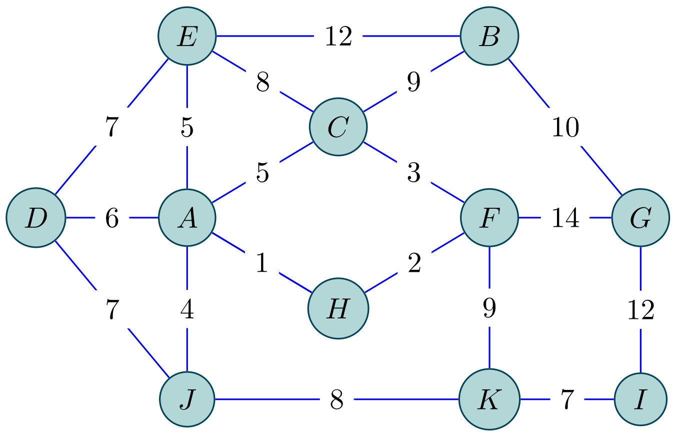


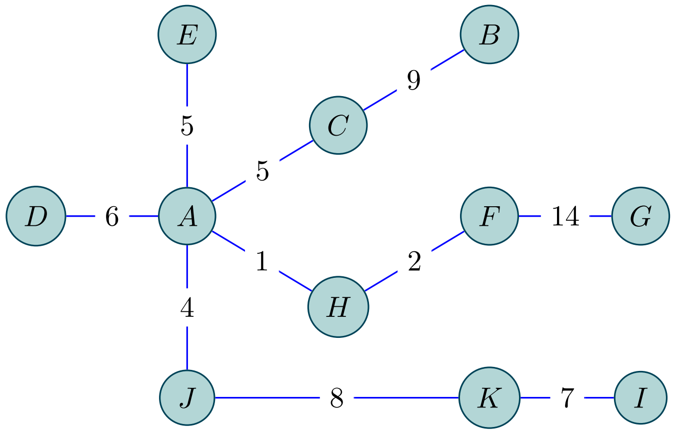


Dijkstra's Shortest Path Algorithm (4 points)

Given the following graph, complete the table showing the result of Dijkstra's shortest path algorithm called on node `A`. Fill in the cost of the shortest path to each node and the order in which the algorithm marks its path cost as "known" (i.e., label with 1, 2, 3, etc.). Node As cost has been labeled 0 and its order 1 since it is the start node. If there is a tie in ordering, sort alphabetically.

node	cost	order
A	0	1
B	14	9
C	5	5
D	6	7
E	5	6
F	3	3
G	17	10
H	1	2
I	19	11
J	4	4
K	12	8





Blank sheet for extra work.