CS61BL Final Review

1 Heaps of Fun

(a) Consider the following array representation of a min-heap.

i	0	1	2	3	4	5	6	7	8	9	10	11
a[i]	*	A	C	P	M	G	Q	X	Z	T	L	O

Show the array after performing each of the following operations on the *original* min-heap.

1. Insert `E`.

a[i]	*												

2. Delete the minimum.

a[i]	*												

(b) Which of the following are advantages of a heap (with a resizing array) over a sorted linked list?

- expected time for `insert` is lower
- expected time for `delMin` is lower
- expected storage cost is lower
- `insert` has lower worst-case runtime
- `delMin` has lower worst-case runtime
- `max` has lower worst-case runtime

(c) What is the size of the largest binary min-heap in which the fifth largest element is a child of the root? *Assume the heap has no duplicate elements.*

2 SixtyOnePQ

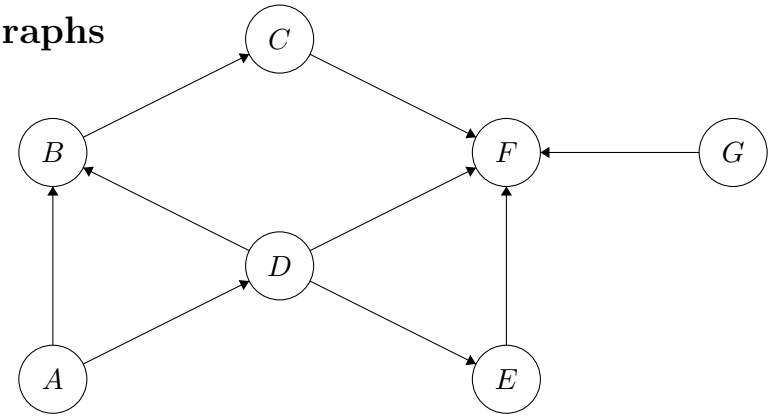
You have been hired by Alan to help design a priority queue-like data structure for Kelp ordered on the timestamp of each `Review`. This data structure needs to support the following operations:

- `insert(Review r)` a `Review` in $O(\log N)$ time.
- `edit(int id, String body)` any one `Review` in $\Theta(1)$ time. (Timestamp does not change.)
- `sixtyOne()` — return the sixty-first latest `Review` in $\Theta(1)$ time.
- `pollSixtyOne()` — remove and return the sixty-first latest `Review` in $O(\log N)$ time.

Explain how you would implement the required functionality, using one or more data structures that we have seen in class. Write pseudocode for each of the operations listed above. You may assume that $N > 61$ and omit all checks for smaller N.

Your implementation should support finding the k^{th} smallest item with an asymptotic runtime independent of k. That is, it should be possible to change 61 to another constant while maintaining the asymptotic runtime bound. For instance, we should be able to find the median by setting $k = \frac{N}{2}$.

3 Graph-Graphs



(a) Show the adjacency matrix and adjacency list representations of the above graph.

(b) With an **adjacency list**, how long does it take to check for the existence of an edge between two vertices? Give the tightest possible asymptotic bound in terms of $|V|$ and $|E|$.

(c) How long does the same operation take with an **adjacency matrix**?

(d) Give the DFS preorder, DFS postorder, and BFS traversals starting from vertex A as well as the DFS postorder topological sort. Break ties alphabetically.

`DFS preorder` | | | | | | | |

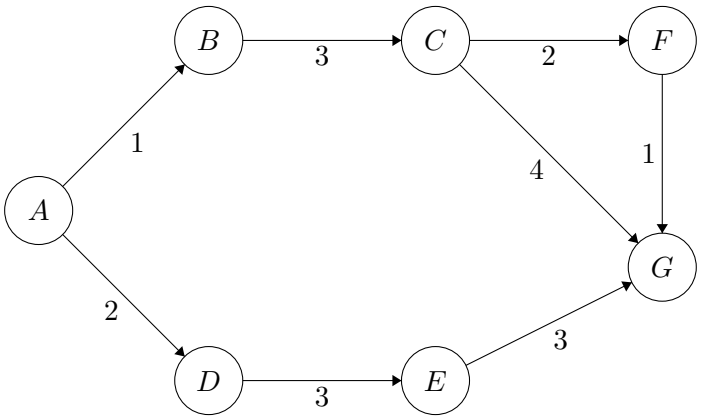
`DFS postorder` | | | | | | | |

`BFS` | | | | | | | |

`Topological` | | | | | | | |

4 Circle the Mascot

$edge(A, B) = 1$ $\quad h(A, G) = 8$
$edge(B, C) = 3$ $\quad h(B, G) = 6$
$edge(C, F) = 2$ $\quad h(C, G) = 5$
$edge(C, G) = 4$ $\quad h(F, G) = 1$
$edge(F, G) = 1$ $\quad h(D, G) = 6$
$edge(A, D) = 2$ $\quad h(E, G) = 3$
$edge(D, E) = 3$
$edge(E, G) = 3$



(a) Run Dijkstra's Algorithm to find the shortest paths from A to every other vertex.

(b) Give the path A* Search would return given the heuristic above and A as the start and G as the goal.

(c) The above heuristic is *inadmissible*: the heuristic returns a greater length than the actual remaining distance needed to reach the goal. Determine the possible range for each inadmissible heuristic value so that the heuristic becomes admissible.

(d) Is it true that, assuming every vertex is reachable from a given source, Dijkstra's Algorithm always finds a shortest path from that source to every vertex?

(e) Given an undirected, positively-weighted graph $G = (V, E)$, a set of start vertices S, and a set of end vertices T, describe an efficient algorithm that returns the shortest path starting from any one vertex in S and ending at any one vertex in T.

5 Sort This Out

For each of the following scenarios, choose the best sorting algorithm and explain your reasoning.

(a) Sorting a list created by first taking a sorted list of N elements and then swapping N pairs of adjacent elements.

(b) Sorting a list of everyone who took the US census by last name.

(c) Sorting a list on a machine where swapping two elements is much more costly than comparing two elements (and you want to do the sort in place).

(d) Sorting a list so large that not all of the data will fit into memory at once. As is, at any given time most of the list must be stored in external memory (on disk), where accessing it is extremely slow.

Each of the following sequences represent an array being sorted at some intermediate step. Match each sample with one of the following sorting algorithms: **insertion sort, selection sort, heap sort, merge sort, quicksort, LSD radix sort, MSD radix sort**. The original array is below.

```
5103 9914 0608 3715 6035 2261 9797 7188 1163 4411
```

(a)
```
5103 9914 0608 3715 2261 6035 7188 9797 1163 4411
0608 2261 3715 5103 6035 7188 9797 9914 1163 4411
```

(b)
```
0608 1163 5103 3715 6035 2261 9797 7188 9914 4411
0608 1163 2261 3715 6035 5103 9797 7188 9914 4411
```

(c)
```
9797 7188 5103 4411 6035 2261 0608 3715 1163 9914
4411 3715 2261 0608 1163 5103 6035 7188 9797 9914
```

(d)
```
2261 4411 5103 1163 9914 3715 6035 9797 0608 7188
6035 5103 1163 7188 2261 4411 0608 3715 9797 9914
```

(e)
```
5103 0608 3715 2261 1163 4411 6035 9914 9797 7188
0608 2261 1163 3715 5103 4411 6035 9914 9797 7188
```

(f)
```
0608 5103 9914 3715 6035 2261 9797 7188 1163 4411
0608 2261 3715 5103 6035 9914 9797 7188 1163 4411
```

6 Disjoint Sets & Kruskal's Algorithm & ... Regex

(a) Give the tightest asymptotic bound on the height of a weighted quick union tree in terms of N, the number of elements. Provide a $\Theta(\cdot)$ bound if possible, else give both $O(\cdot)$ and $\Omega(\cdot)$.

(b) Is it true that, given a graph G, if we add some constant k to every edge weight, G's minimal spanning tree(s) remain unchanged?

(c) Assume we are using disjoint sets with path compression. How many calls to `find` need to be made in order for each node to be directly connected to the root node?

(d) In terms of runtime, what is the worst way to place the integers 1, 2, 3, 4, 5 into the same disjoint set? Give an answer in the form of a series of calls to the `connect` method.

(e) What are the shortest and tallest heights possible for a Quick Union with N elements? What does this mean for the best and worst-case runtime for `isConnected` and `connect`?

(f) What is the shortest and tallest height possible for a Weighted Quick Union with N elements? What does this mean for the best and worst-case runtime for `isConnected` and `connect`?

(g) Mark all of the strings matched by the regular expression `((C|S)* 61*BL+)`.

`C 6B` `CS61B` `CSS 6BL` `CS6BL` `S 61BL` `CS 61BL`

(h) Write a regular expression that matches any binary string of length 8, 16, 32, or 64.

(i) A certain type of bracketed list consists of words composed of one or more lower-case letters alternating with unsigned decimal numerals and separated by commas, as in these examples:

`[]` `[wolf, 12, badger, 11]` `[dog,10, cat]`

Commas may be followed (but not preceded) by blanks, and the list may have odd length (ending in a word with no following numeral). There are no spaces around the `[]` brackets. Write a regular expression that matches such lists.