# 1

1. [k4 k3 k2 k1] [k4 k2 k3 k1] [k4 k3 k1 k2]
2. (0 (<k1) + 0 (<k2) + 1 (<k3) + 3 (<k4) + 6 (>k4))/ 3(heaps) \* 5(position) = ⅔

# 2

1. The height of the tree.
2. A result.
3. n-2, otherwise we don’t know the ordering between the two largest element and the rest of the elements

# 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | Adj Matrix | Adj List | Inv Adj List | Edge List |
| Edge query | 1 | min(|V|,|E|) | min(|V|,|E|)| | |E| |
| Length 2 path | |V| | min(|V|^2,|E|) | min(|V|^2,|E|) | |E| |
| In neib | |V| | min(|V|,|E|) | min(|V|,|E|) | |E| |
| Self loop | |V| | min(|V|,|E|) | min(|V|,|E|) | |E| |
| Edge count | |V|^2 | |V| | |V| | 1 |

# 4

1.  The other tree is the same
2. 
3. 

# 5

1. 0 1 1 1, 1 0 1 1, 1 1 0 1, 1 1 1 0
2. n
3. the first n-1 rows has at most n-2 edges
4. n^2
5. (½)^(n-1)
6. 1 - (½)^(n-1)
7. Let p = (½)^(n-1); so it is n \* (1 + p + (1-p)^1p + (1-p)^2p + … + (1-p)^(n-1)p )
   1. Where 1, p, (1-p)^kp is the probability that we examine any line

# 6

1. Hash table
2. THIS IS NOT A SOLUTION: write any hash table lookup, assume SUHA
3. Balanced binary search tree / AVL tree
   1. Augment each element with the size of its subtree
4. Add product adds the element to both the hash table and BST/AVL and recompute the augmented data when we do insertion
5. TopRated calls TopRatedImpl with the Root Node and r

TopRatedImpl(N, r)

If N is Nil: return 0

If N.key >= r: # Assume the size field of a Nil node is 0

return N.left.size + 1 + TopRatedImpl(N.right, r)

else:

return TopRatedImpl(N.left, r)

# 7

1. Amortized time per append: O(1), Charge for ith append 3/2
   1. The later ⅔ half of the array need to cover the cost of a resize, so charge them 3/2 each
2. O(n), n/263 The last 263 element of the array need to cover the cost of a resize
3. Accounting method: the last ⅓ element of the array need to pay for a full array resize, so charge them 3 each.
   1. The invariant is that the total charges plus three times the number empty slot is at least the size of the array
   2. USE INDUCTION TO PROVE IT

# 8

It requires a neight idea, so we have words, and something called intermediate words.

So PHYSICS is connected to \_HYSICS / P\_YSICS / … / PHYSIC\_ .

1. The graph consists all the words and their corresponding intermediate words. Each word has 7 edges, which is to its intermediate words. The edges are undirected.
2. We can use a hashtable (or a huge array 26^7 is not too much space for a modern computer)
3. NOT FULL SOLUTION: For each word we have 7 intermediate words and 7 edges, so the number of vertex and edges are bound (8n vertex + 7n edges), and hash table lookup for vertex is O(1).
4. Use BFS, take the path and pick all the words from the path (not intermediate words), EXPLAIN why BFS find the shortest path: the actual path length is ½ of the BFS path, and BFS always finds the shortest path.