

Computer Science 410, Spring 1999: Final Exam Review

1. Suppose $T(n) = T(\lfloor an \rfloor) + T(\lfloor bn \rfloor) + \lfloor cn \rfloor$, where $a + b < 1$. Show that $T(n) = O(n)$.
2. Do Problem 10.3-5 in the text.
3. Given an adjacency list representation of a directed graph $G = (V, E)$, write algorithm that runs in time $O(|V| + |E|)$ that returns two arrays In and Out of length $|V|$ such that $In[v]$ is the indegree of v and $Out[v]$ is the outdegree of v .
4. You have a hash table of size $m = 11$ and hash functions h_1 and h_2

$$h_1(k) = k \bmod m$$

$$h_2(k) = 1 + (k \bmod (m - 1)).$$

- (a) Show the contents of the hash table after inserting, in order, the keys 10, 22, 32, 4, 15, using chaining and h_1 only.
 - (b) Show the contents of the hash table after inserting, in order, the keys 10, 22, 32, 4, 15, using **open addressing with double hashing**, and $h_1(k)$ and $h_2(k)$ above, as hash functions.
5. Assume you know that there will be no more than n different keys, and want to use open addressing with double hashing. What table size and what functions you would use?
 6.
 - (a) State the *heap property*. (Assume we want the biggest element at the root of the heap.)
 - (b) Construct a heap containing the following keys: 1,2,9,11,12, 15.
 - (c) What happens to the in part (b) after an EXTRACT-MAX.
 - (d) Give $O(\log n)$ algorithms (either in understandable pseudocode or explain how it works) to *increase* the key of an element, and to *decrease* the key of an element in a heap. (Here n denotes the number of elements in the Heap.) Recall that we need DECREASE-KEY in Dijkstra's algorithm and Prim's algorithm.
 7. Assume e is the unique least expensive edge in the graph. Explain why e is in **all** minimum spanning trees.
 8. Mr. Phillips discovered a new way of sorting numbers. His sorting algorithm does comparisons and rearranging of data only. He claims that his algorithm sorts n numbers in $O(n)$ time, and would like to sell your company the right to use the algorithm for a mere \$5,000,000. Should you pay him? Why or why not?

9. Suppose you have an application of priority queues where you know the that there can be only k different priorities, called priorities $1, \dots, k$. You know that there will be a large number N of items in your priority queue, where N is much bigger than k , so many items will have the same priority.
- (a) Suppose you use a standard heap-based priority queue and put the N elements in this priority queue. What is the worst case running time of `delMax` and `insert` in this implementation?
 - (b) Design a data structure for this application that allows the `Insert` and `delMax` operations to run in $O(\log k)$ time. You may use any data structure that we learned about without writing code for it, but be careful in explaining any modification you must make for this problem. Explain clearly how you do `insert` and `delMax`. You may use any data structure operations that we learned about without writing code for them.

10. **Implementing a data base operation:**

A database often consists of data which is in the form of records. For example, the ACM (Association of Computing Machinery) membership database might include lists of records of the members of the different subgroups of ACM. (These subgroups are called SIGs, or Special Interest Group.) Here's an example of a very small database of this form:

SIGGRAPH	SIGCOMM
Jones	Wilson
Chang	Jones
Markowitz	Smith
Brown	Chang

A standard database operation is the *join* operation: Given two lists of records, find their common members. So, for example, given the database above, the join of the SIGCOMM members and the SIGGRAPH members is the list (Jones, Chang).

In this problem, we compare methods which implement this operation.

Assume for simplicity that we consider lists of numbers (rather than items with keys) and the lists have no duplicates. One list is stored in an array `COMM` of length N , and the other list in an array `GRAPH` of length M , and we want to have the numbers that occur in both listed in an array `SIG`. Assume $N > M$. **Give the expected running times of each of the methods using the $O(\cdot)$ notation and the parameters N and M .** Explain your answer.

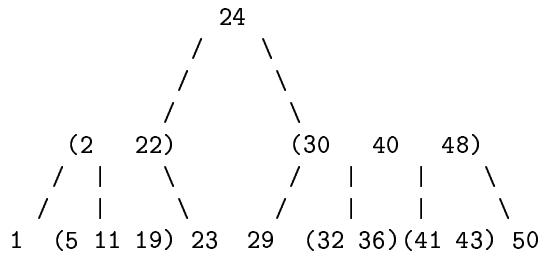
- (a) Nested loops going through the arrays `GRAPH` and `COMM`.

```
int i,j,k=0;
for(i=0; i < N; i++)
  for(j=0; j<M; j++)
    if( GRAPH[i] == COMM[j])
      {SIG[k] = COMM[j]; k++;}
```

- (b) Sort the items on the shorter list `GRAPH`, and then go though the unsorted list `COMM` (a single for loop) and use binary search for each of the numbers in the unsorted `COMM` list to check if they also occur on the `GRAPH` list; if they do, copy them into the `SIG` array.

- (c) Select a good hash function, h , and a table size, L , close to M . Use this hash function h to hash the elements of the shorter list, **GRAPH**, into a new table, **H-GRAPH**, using chaining to resolve collisions. Now, go through the other list, **COMM**, (with a single **for** loop). Use the search method of hashing to look for each of the numbers in the **COMM** list to check if they also occur on the **H-GRAPH** hash table; if they do, copy them into the **SIG** array.
- (d) Build a binary search tree from the items in the shorter list **GRAPH**. Now, go through the other list, **COMM**, (with a single **for** loop). Use the search method of BST to look for each of the numbers in the **COMM** list to check if they also occur in the BST; if they do, copy them into the **SIG** array. Is it better in this method to build the BST from the shorter list, or should we build the BST using the longer list and then search for each element of the longer list in this BST? Explain your answer.
11. Assume you have to sort the pixels of a 5,000 by 5,000 pixel image by color intensity, and the color intensity of a pixel is an integer in the range of $0, \dots, 127$. **What sorting method would you use?** Note that in processing pictures, speed is very important. (For example, you might want to process 10-20 pictures per second, if you want to be able to process a movie as you play it.) **Explain your choice** and **give the running time** of your method using the $O(\cdot)$ notation with the parameters of $N = 25,000,000$: the number of pixels to be sorted and $R = 128$: the number of different color intensities.
12. Given an undirected graph G on n nodes and m edges, where each node has degree d (note, $n \times d = 2m$). Assume G is given by an adjacency matrix A .
- (a) Given a node r of G , describe (either in understandable pseudocode or in clear English) how to list all nodes that are reachable from r on paths of length at most 2 (i.e., using at most 2 edges). List every node at most once. Full credit will be given for algorithms that run in $O(nd)$ time; partial credit for an $O(n^2)$ algorithm. Explain why your algorithm works and what its running time is.
- (b) Solve the question in the previous part if the graph G is given by adjacency list. Full credit will be given for algorithms that run in either $O(n + d^2)$ or $O(d^4)$ time. Explain why your algorithm works and what its running time is.
13. Give an algorithm that takes as input a graph $G(V, E)$ (using the adjacency-list representation) and an edge $(u, v) \in E$ and decides if G has a cycle that goes through edge (u, v) , and if so, outputs the cycle. The algorithm should run in time $O(|V| + |E|)$. You may use any algorithm from class without writing it out. You may also use a modified version by clearly explaining what needs to be modified. Explain why your algorithm works and what its running time is.
14. Consider an undirected graph $G = (V, E)$ with nonnegative weights $w(i, j) \geq 0$ on its edges $(i, j) \in E$. Let s be a node in G . Assume you have computed the shortest path tree and the minimum spanning tree. Suppose we change the weights on every edge by adding the same constant $c \geq 0$ to each of them. The new weights are $w'(i, j) = w(i, j) + c$ for every $(i, j) \in E$.
- (a) Would the minimum spanning tree change due to the change in weights? Give an example where it changes, or explain why it cannot change.
- (b) Would the shortest path tree change due to the change in weights? Give an example where it changes, or explain why it cannot change.
15. Do problem 23.3-9 in the text.
16. Do problem 23.4-3 in the text. (Hint: they actually give you the algorithm in the text. All you have to do is argue that it runs in time $O(|V|)$.)

17. Do problem 25.2-4 in the text.
18. Run Dijkstra's algorithm on the graph in Figure 25.2, using v as the source.
19. Run Bellman-Ford on the graph in Figure 25.7, using x as the source, and changing the weight of (y, v) to 5.
20. List the edges in the MST computed by Kruskal's algorithm (in the order that they're added) on the graph in Figure 24.4 in the text, if you change the weight of the edge (h, i) to be -3 and the weight of (e, f) to be 4. Now do the same for Prim's algorithm on the same graph, starting with vertex e . (You should break ties in lexicographic order.)
21. The following picture represents a B-tree with $t = 2$ (i.e., a 234-tree).



- (a) Draw an equivalent red-black-tree.
 - (b) Draw the 234-tree that results from inserting 31 into the original 234-tree.
 - (c) Draw the 234-tree that results from deleting 24 from the original 234-tree.
22. Consider an ADT with the following operations: Insert, GetNext, and GetMax. GetNext returns the next item in FIFO (First-in First-out) order and deletes it from the set. Thus, if only Insert and GetNext are used then the ADT acts like a queue. GetMax returns the item with the maximum key value and deletes it from the set. Thus, if only Insert and GetMax are used, then the ADT acts like a priority queue. Describe a data structure that implements this ADT efficiently. How long does each operation take?